STA-3900
Master thesis in statistics

Applying the ICM algorithm for separating pictures consisting of two main parts, background and object
Application to fMRI recordings and mole pictures

Are Larsen

May 2009

FACULTY OF SCIENCE
Department of Mathematics and Statistics
University of Tromsø

Acknowledgements

Thanks to my supervisor Professor Fred Godtliebsen for useful advice and inspiration.

Thanks to Torgil Vangberg at the University Hospital in Tromsoe for providing me with fMRI recordings and Lena Ringstad Olsen for cooperation in the first year of my master study. Thanks to Kajsa Moellersen at the Norwegian Center for Telemedicine for providing me with photographs of moles and software for initial processing of the pictures.

Contents

Chapter 1

Introduction

A common task in the subject of mathematical statistics with many applications is distinguishing between objects belonging to some kind of predefined sets. Several words relate to basically the same kind of problem: Detection, separation, classification, allocation, hypothesis testing. Which word is used depends on the setting. Detection is normally used when the problem is to extract information from a signal. Separation is often used when the problem is to sort some kind of objects or observations into some predefined sets. Classification or allocation usually refers to the task of assigning new objects to predefined classes. Hypothesis testing is used about the problem of testing the reliability of a statement in comparison to alternative statements. The theory of hypothesis testing form the basis for the other concepts.

In this text the above mentioned tasks is limited to distinguishing between *two* predefined sets of objects or observations. Often hypothesis testing includes the use of multivariate analysis, but in this text the test statistic is restricted to a single dimensional variable. The problem is to select between either of two hypothesis. We assume that the true value of our test statistic is either of *two* possible values. So the use of hypothesis testing is simplified to a binary hypothesis test, testing a simple hypothesis against another simple hypothesis. The Neyman-Pearson lemma tells how such a test is done optimally by using a likelihood ratio test.

Further the problem treated in this text is about separating objects in a picture. By examining pictures it is found that not all configurations of the voxel values are equally probable. It is of course not feasible to completely define this multivariate distribution. But simplifications lead to useful approximations. The concept of a Random Markov Field defining a set of probabilities $\{p(\mathbf{x})\}$ for each picture element individually is used together with the use of computer algorithms. The density functions $\{p_i(x_i|x_{\partial i})\}, i \in S$ are given as conditionally dependent upon the set of picture elements defined as the neighbourhood for each voxel. Introducing the conditionally dependent probability functions $\{p_i(x_i|x_{\partial i})\}$ reduces errors by providing individual values of the prior probabilities for the picture elements, dependent on the values of the picture

elements in the neighbourhood.

This text concentrates on doing separation of *picture* elements in pictures containing *two* predescribed populations. These populations are often referred to as object and background. The applications are numerous. The problem might be a need to remove random noise to ease further reading of the picture by humans. Or it might be to convert the information given by the picture to a form convenient for further automated processing. Cost savings can be obtained by using computer programs for initial processing of pictures, sorting out pictures for further study by humans. Another application is using cameras to detect objects of a given shape.

In this text the ICM algorithm (Iterated Conditional Modes is used for computer processing of the pictures combining the assumption of conditional independence of the observed values of the picture and the assumption of a Random Markov Field.

This text focuses on two examples of pictures for application:

- 3-dimensional fMRI (functional Magnetic Resonance Imaging) recordings used in medicine and research
- 2-dimensional photos of moles used in medicine as the initial step in the process of detecting melanoma.

Chapter 2

Basic concepts for the separation of pictures consisting of two main parts

2.1 Hypothesis test for detecting the value of the test statistic

Using this model the problem of detecting if a picture element shows the background or the object is a binary hypothesis test:

H_0: element shows background $\quad versus \quad$ H_1: element shows object

To perform this hypothesis test, a test statistic or discriminant has to be defined. If the expected value of d is μ_0 for the background and μ_1 for the object, then our hypothesis test is

$$H_0 : d = \mu_0 \qquad versus \qquad H_1 : d = \mu_1 \tag{2.1}$$

This is just a binary hypothesis test, in this case testing a simple hypothesis against another simple alternative. In the following we briefly review some results of interest for this kind of test.

To choose between H_0 and H_1 we have to select a threshold for $\hat{d}$. Obviously we want to have small probabilities for both Type I and Type II Errors (H_0 incorrectly rejected or accepted respectively). For a test of a given size α we want maximum power. There is a well known theorem which tells how to choose the threshold to obtain maximum power for a given Type I error rate:

(**Neyman−Pearson Lemma**) *Consider testing* $H_0 : \theta = \theta_0$ *versus* $H_1 : \theta = \theta_1$, *where the pdf or pmf corresponding to* θ_i *is* $f(\mathbf{x}|\theta_i), i = 0, 1$, *using a test with rejection region* R *that satisfies*

$$\begin{aligned} \mathbf{x} \in R \quad & if \quad f(\mathbf{x}|\theta_1) > kf(\mathbf{x}|\theta_0) \\ & and \\ \mathbf{x} \in R^c \quad & if \quad f(\mathbf{x}|\theta_1) < kf(\mathbf{x}|\theta_0) \end{aligned} \tag{2.2}$$

for some $k \geq 0$, *and*

$$\alpha = P_{\theta_0}(\mathbf{X} \in R) \tag{2.3}$$

Then
a. *Any test that satisfies* (2.2) *and* (2.3) *is a uniformly maximum power (UMP) level* α *test.*
b. *If there exists satisfying* (2.2) *and* (2.3) *with* $k > 0$, *then every UMP level* α *test satisfies* (2.2) *except perhaps on a set* A *satisfying* $P_{\theta_0}(\mathbf{X} \in A) = P_{\theta_1}(\mathbf{X} \in A) = 0$.

2.2 Applying the Neyman-Pearson Lemma

The Neyman-Pearson Lemma tells that the ratio

$$l(\mathbf{x}) = \frac{L(\theta_1|\mathbf{x})}{L(\theta_0|\mathbf{x})} = \frac{f(\mathbf{x}|\theta_1)}{f(\mathbf{x}|\theta_0)}$$

gives a UMP test. If $l(\mathbf{x}) > 1$ then $\theta = \theta_1$ is the more likely value for θ. Otherwise $\theta = \theta_0$ is the more likely value.

Applying this to our model we substitute $\hat{d}$ for $\mathbf{x}$, $d = \mu_0$ for θ_0, and $d = \mu_1$ for θ_1:

$$l(\hat{d}) = \frac{L(d = \mu_1|\hat{d})}{L(d = \mu_0|\hat{d})} = \frac{f(\hat{d}|H_1)}{f(\hat{d}|H_0)}$$

Using a binary hypothesis test the threshold is given by

$$f(\hat{d}|H_1) = f(\hat{d}|H_0) \tag{2.4}$$

The density functions $f(\hat{d}|H_0)$ and $f(\hat{d}|H_1)$ are distributions symmetric around their mean with the same variance. $l(\hat{d}) > 1$ corresponds to $\hat{d} > \frac{\mu_1 - \mu_0}{2}$ if $\mu_1 > \mu_0$ or $\hat{d} < \frac{\mu_1 - \mu_0}{2}$ if $\mu_0 > \mu_1$. Thus choosing $\frac{\mu_1 - \mu_0}{2}$ as the threshold value gives equal probabilities for correctly detecting the background or the object.

Of course other choices of the parameter α, giving another value for the threshold, can be done to optimize the test against other requirements. But in this section we restrict our treatment to choosing the threshold to maximize the probability of correctly classifying a picture element.

2.3 Bayes' theorem

The Bayesian theorem or Bayesian formula is elementary in statistics. We state it here because of its importance in image restoration. It is based upon the definition of conditional probability, i.e. the probability when the sample space is restricted to the outcomes when some given event is true. If X and Y are two events and $X \cap Y$ is their joint occurence

$$P(X|Y) = \frac{P(X \cap Y)}{P(Y)} \qquad P(Y|X) = \frac{P(X \cap Y)}{P(X)}$$

Combining these two expressions we obtain

$$P(X|Y) = \frac{P(Y|X) \cdot P(X)}{P(Y)}$$

It is used for "inverting" conditional probabilities. In image restoration we want to estimate the true image. If we know the probability of the observed image Y given the true image X, which is the case if we know the distribution of the noise, we can obtain an expression for the probability of the true image by "inverting" this conditional probability. $P(X)$ is the probability for X if no observation is done and is referred to as the prior probability. $P(Y)$, in Bayesian terms named the "total probability", is the probability of all observed scenes regardless of the true picture. In image restoration it is obviously difficult to express the latter two probabilities. But in restoration the problem is to estimate X, most often done by finding the value of X which maximizes $P(X|Y)$, which can be done without knowledge about $P(Y)$. The remaining problem is $P(X)$.

2.4 Incorporating prior probability in the test

If we assume that a picture element has probabilities $P(d = \mu_1)$ of showing the object and $P(d = \mu_0)$ of showing the background, then we can use Bayes' theorem

$$P(d|\hat{d}) = \frac{f(\hat{d}|d) \cdot P(d)}{f(\hat{d})}$$

to improve the chances of correctly classifying a picture element. For the likelihood ratio we obtain

$$\frac{P(d = \mu_1|\hat{d})}{P(d = \mu_0|\hat{d})} = \frac{\frac{f(\hat{d}|d=\mu_1) \cdot P(d=\mu_1)}{f(\hat{d})}}{\frac{f(\hat{d}|d=\mu_0) \cdot P(d=0)}{f(\hat{d})}} = \frac{f(\hat{d}|H_1) \cdot P(H_1)}{f(\hat{d}|H_0) \cdot P(H_0)}$$

If $P(H_0) = p$, then $P(H_1) = 1 - p$. After inserting this parameter we obtain for the likelihood ratio

$$\frac{P(H_1|\hat{d})}{P(H_0|\hat{d})} = \frac{f(\hat{d}|H_1) \cdot (1-p)}{f(\hat{d}|H_0) \cdot p} = l(\hat{d}) \cdot \frac{1-p}{p} \tag{2.5}$$

To determine if a picture element is showing the object or the background the threshold should be chosen such that the picture element is regarded as showing the object if this ratio is greater than 1 for $\mu_1 > \mu 0$ or less than 1 for $\mu_0 > \mu_1$, i.e. the threshold chosen by taking the prior probability into account is the value of $\hat{d}$ for which

$$f(\hat{d}|H_1) \cdot (1-p) = f(\hat{d}|H_0) \cdot p \tag{2.6}$$

2.5 Error probabilities

If the detection is done by choosing the threshold with respect to minimizing the probability of error regardless of the contents of a picture element, i.e. by not taking prior probabilities into account, the error probability is equal to the size α of the hypothesis test given H_0 and equal to $1-\beta$ given H_1. (β is the power of the test). If $\mu_1 > \mu_0$ the error rates are

$$P(error|H_0) = \int_{\frac{\mu_1-\mu_0}{2}}^{+\infty} f(x|H_0)\mathrm{d}x = \alpha \tag{2.7}$$

$$P(error|H_1) = \int_{-\infty}^{\frac{\mu_1-\mu_0}{2}} f(x|H_1)\mathrm{d}x = 1-\beta \tag{2.8}$$

If $\mu_0 > \mu_1$ the lower and upper boundary in the first and second expression respectively must be substituted by $\frac{\mu_0-\mu_1}{2}$.

If the prior probability of the picture element showing the background is taken into account, the average error probability is improved. The probability of correctly detecting a picture element containing its most probable contents is improved but the probability of a detection error if the picture element is not containing its most probable contents increases. The probabilities of error given H_0 or given H_1 are by definition given by α and $1-\beta$ respectively. But now these probabilities differ due to the choice of threshold. If $\hat{d}_{thres}$ is the value of $\hat{d}$ given by (2.6), then the error probabilities are given by

$$P(error|H_0) = \int_{\hat{d}_{thres}}^{+\infty} f(x|H_0)\mathrm{d}x = \alpha \tag{2.9}$$

$$P(error|H_1) = \int_{-\infty}^{\hat{d}_{thres}} f(x|H_1)\mathrm{d}x = 1-\beta \tag{2.10}$$

Error rate when prior probability is taken into account:

$$\begin{aligned} P(error) &= P(error|H_0) \cdot P(H_0) + P(error|H_1) \cdot P(H_1) \\ &= \alpha p + (1-\beta)(1-p) \end{aligned} \tag{2.11}$$

2.6 Probability density function for our picture model

So far no assumptions has been made about the shape of the distributions $f(x|H_0)$ and $f(x|H_1)$. In the continuation we assume that the distributions are gaussian or normal in shape. Due to the Central Limit Theorem it is usually the case that noise has a normal distribution. Furthermore the test statistic is often obtained by summing and/or subtracting independent and identically distributed random variables. In these case the approximation to a normal distribution will be even better. If the sum contains several terms, test statistics showing a close approximation to a normal distribution can be obtained even from noise distributions far from normal in shape.

The test statistic $\hat{d}$ is modelled as the sum of the true value of d and an error term ε

$$\hat{d} = d + \varepsilon \tag{2.12}$$

d takes either of the values μ_0 or μ_1 and $\varepsilon \sim N(0, \sigma^2)$ is gaussian noise.

If the variance of the test statistic depends upon whether the picture element shows the background or the object, the model can be modified to allow for different magnitudes of the noise:

$$\hat{d} = d_i + \varepsilon_i, \qquad \text{given } H_i, i = 0 \text{ or } 1, \qquad \varepsilon_i \sim N(0, \sigma_i^2) \tag{2.13}$$

If we let X be a Bernoulli random variable, $X = 0$ if the picture element is showing background with probability p, $X = 1$ if the picture element is showing the object, we have

$$\begin{aligned} \hat{d}|X &\sim \text{normal}(\mu_X, {\sigma_X}^2) \\ X &\sim \text{Bernoulli}(1-p) \end{aligned}$$

This is a mixture distribution, in statistical terms a normal-Bernoulli hierarchical model with probability density function

$$f(\hat{d}; {\sigma_0}^2, {\sigma_1}^2, \mu_0, \mu_1, p) = \frac{p}{\sqrt{2\pi{\sigma_0}^2}} e^{-\frac{(\hat{d}-\mu_0)^2}{2{\sigma_0}^2}} + \frac{1-p}{\sqrt{2\pi{\sigma_1}^2}} e^{-\frac{(\hat{d}-\mu_1)^2}{2{\sigma_1}^2}} \tag{2.14}$$

2.7 Estimation of parameters in the mixture distribution

If $\hat{d}_i, i = 1 \ldots n$ are n estimates of $\hat{d}$, the likelihood function is

$$\begin{aligned} L({\sigma_0}^2, {\sigma_1}^2, \mu_0, \mu_1, p; \hat{\mathbf{d}}) &= f(\hat{\mathbf{d}}; {\sigma_0}^2, {\sigma_1}^2, \mu_0, \mu_1, p) \\ &= \prod_{i=1}^{n} \left(\frac{p}{\sqrt{2\pi{\sigma_0}^2}} e^{-\frac{(\hat{d}_i-\mu_0^2}{2{\sigma_0}^2}} + \frac{1-p}{\sqrt{2\pi{\sigma_1}^2}} e^{-\frac{(\hat{d}_i-\mu_0)^2}{2{\sigma_1}^2}} \right) \end{aligned} \tag{2.15}$$

Even for this very simple mixture distribution it is necessary to use numerical methods for estimating MLEs of ${\sigma_0}^2$, ${\sigma_1}^2$, μ_0, μ_1 and p. (See "Robert and Casella: Monte Carlo Statistical Methods", second edition, page 4 and 11 for a brief introduction.)

Chapter 3

Applying our simple model

As an illustration of our model to make things clearer, we try using it on a simulated data set.

3.1 Simulating the probability model

Simulating the normal-Bernoulli hierarchical distribution is quite easy using any statistical program package. The Bernoulli part of the hierarchical tree can be simulated by using a command simulating a uniformly distributed random variable on the interval (0, 1). The normal part can be simulated using a command simulating a standard normal distributed random variable.

3.2 Estimation of parameters

The values of all five parameters might be unknown, but here we consider a simplified example. We assume that $\mu_0 = 0$ and $\sigma_1 = \sigma_0$. We need to estimate the three parameters p, μ and σ. See figure 3.1.

The density function is a sum of two normally distributed peaks having the same variance. The position of the peak with the smallest mean, in this case the larger, is known. A good initial estimate of the standard deviation σ and the parameter p can be found by looking at the leftmost half of the leftmost peak in the plot of the function. By mirroring the peak around is mean (0 in this case), standard functions for estimating variance can be used. p is estimated by just counting the values comprising the left half of the larger peak and the total number of samples making the histogram. This method gives good estimates for our model due to the known position of the peak and the size of this peak compared to the remaining peak.

Obtaining a good estimate for the parameter μ is not quite as simple because the second term in the mixture distribution usually is made from fewer samples. But already having good estimates for σ and p makes it easy to use MLE estimation after insertion into the likelihood function. A routine for finding an

extremal value of a function of a scalar on a given interval is often included in advanced high level programming languages. It turns out that it is better to use the loglikelihood function to avoid numerical underflow in the computer.

After obtaining these good initial approximations for the parameters a routine for maximum likelihood estimation can be run to get better estimates of all three parameters simultaneously.

3.3 Simulation run

The histogram from a simulation with parameter values $\sigma = 0.5$, $p = 0.95$ and $\mu = 2$ is shown. Then these parameters have been estimated from the simulated data. The given density function for the simulation and the estimated density function (both multiplied by total number of generated values times interval width corresponding to each histogram column) have been plotted. The curves resulting from both the initial course approximation of the parameters and from the final MLE estimation have been drawn for comparison.

As can be seen from the figure the suggested procedure for reducing computation while estimating the parameters performs pretty well for this density function with the mean value for the larger peak known and equal σ's for the two peaks. Only small adjustments to the estimates result when using maximum likelihood to estimate all three parameters. The deviation between the simulated density function and the density function estimated from the simulated data is mainly caused by differing σ's.

3.4 Choosing the threshold for the test

The optimum threshold should of course be calculated from the true values of the parameters. But in our case they are not known, so we have to calculate a threshold value from the estimated parameters.

Figures 3.2 and 3.3 illustrate how the thresholds given by (2.4) and (2.6) are found. Separate histograms of densities for background and object are given to show how the probabilities of Type I and Type II Errors change by different choices of threshold.

3.5 Detection, classification

So far we have considered our problem as separating two populations, background elements and object elements, by using a hypothesis test. We have considered the mixture distribution as the sum of two separate normal distributions. The picture elements has been considered as elements fixed in either of the two possible states. From this point of view we have to decide which normal distribution a picture element exhibits for the test statistic, or discriminant, $\hat{d}$. The same mixture distribution can be used as a density model for the test statistic of a fixed element which can assume either of two states. In the latter case

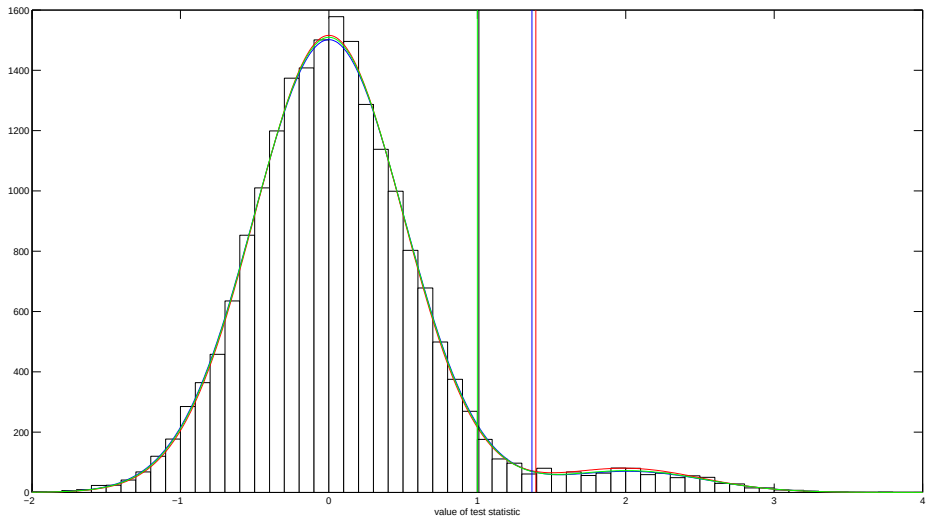

Figure 3.1: Histogram of 20,000 values drawn from the mixture distribution or normal-Bernoulli hierarchical model with $\sigma = 0.500$, $p = 0.950$ and $\mu = 2.000$ (red curve). Working in the opposite direction $\hat{\sigma} = 0.508$, $\hat{p} = 0.956$ and $\hat{\mu} = 2.027$ (blue curve) is found from the proposed procedure for estimating the parameters. Finally the MLEs for all three parameters are obtained, $\hat{\sigma} = 0.505$, $\hat{p} = 0.955$ and $\hat{\mu} = 2.023$ (green curve). The green and the black vertical lines are the thresholds given by an ordinary hypothesis test for detecting picture element state calculated from the parameters used for simulation and from the parameters estimated from the simulated values respectively. Likewise the blue (calculated from simulation parameters) and the red (calculated from parameters estimated from simulated data) vertical lines are the thresholds given by taking prior probability into account. The deviations of $\hat{\sigma}$, $\hat{p}$ and $\hat{\mu}$ all contributes to $\hat{d}_{thres}$ being greater than d_{thres}. The deviation of $\hat{p}$ accounts for half of the difference while the deviations of $\hat{\sigma}$ and $\hat{\mu}$ gives equal contributions approximately.

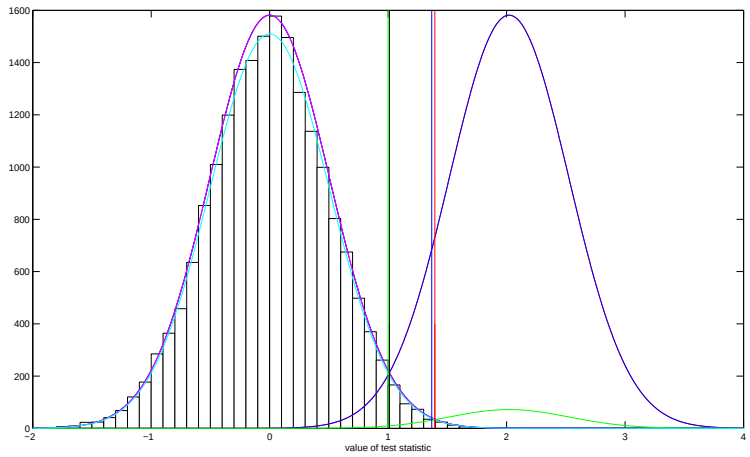

Figure 3.2: The estimated threshold (black vertical line) given by a hypothesis test without prior probabilities taken into account is given by the value of $\hat{d}$ for which the density functions $f(\hat{d}|H_0)$ (magenta) and $f(\hat{d}|H_1)$ (blue) are equal. If prior probabilities are given the threshold (red vertical line) is given by the value of $\hat{d}$ for which $f(\hat{d}|H_0) \cdot p$ (cyan) and $f(\hat{d}|H_1) \cdot (1-p)$ (green) are equal. The green and blue vertical lines are the thresholds given by the parameters used during simulation. The histogram of the data for which the Bernoulli variable in the simulation is 0 (background) is plotted. Test statistic values to the right of the black or red line give Type I Errors without or with prior probability taken into account respectively while performing the test.

the word detection is commonly used. The calculations are totally equivalent regardless of point of view. In either case we have to work with the shape of the histogram of the two summed normal distributions multiplied by the column width and the total number of picture elements making the histogram.

3.6 The Bayesian theorem applied to pictures

Using Bayes' theorem we have

$$P(\mathbf{x}|\hat{\mathbf{d}}) = \frac{P(\hat{\mathbf{d}}|\mathbf{x}) \cdot P(\mathbf{x})}{P(\hat{\mathbf{d}})}, \qquad P(\hat{\mathbf{d}}) = \int_S P(\hat{\mathbf{d}}|\mathbf{x}) \cdot \mathrm{d}P(\mathbf{x}) \tag{3.1}$$

$\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is the true but unknown scene and $\hat{\mathbf{d}} = (\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_n)$ is the observed and hence known test statistic. The elements of $\mathbf{x}$ and $\hat{\mathbf{d}}$ are named x_i and $\hat{d}_i$ for simplicity but they may be the elements of a picture in several dimensions. A reasonable choice for an estimate of the true scene $\mathbf{x}$ is to take $\mathbf{x}$

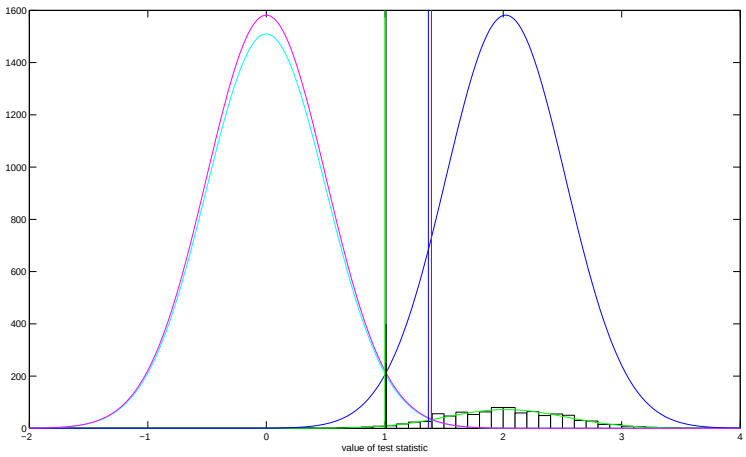

Figure 3.3: The same as the previous figure, but histogram of the values from the simulation corresponding to activated state of picture element (simulated Bernoulli variable equals 1) is plotted. Test statistic values to the left of the black (disregarding prior probability) or red line (prior probability taken into account) give Type II Errors while testing.

to be the value having maximum probability given the observed scene. That is, $\mathbf{x}$ is taken as the value that maximizes $P(\hat{\mathbf{d}}|\mathbf{x}) \cdot P(\mathbf{x})$. $P(\hat{\mathbf{d}}|\mathbf{x})$ is determined by the properties of the noise and usually can be easily formulated. The problem is to find the probability $P(\mathbf{x})$. If this probability function could be given in a tractable way, Bayesian restoration would be easy providing very good results compared to other concepts. Since this in the vast majority of cases is far from the truth it is necessary to use both simplification and approximation.

In practice Bayesian restoration is most often done by considering the state of each picture element individually to greatly simplify matters. Furthermore the function for the prior probability for the state of each picture element, $P(x_i)$, is considered to be a function of a chosen set of nearby picture elements, named the neighbourhood of element i, abbreaviated ∂i. $x_{\partial i}$ is the state of the elements in ∂i. Our test statistic is chosen such that each $\hat{d}_i$ depends on a single x_i. This gives the following expression for Bayesian restoration of picture element i:

$$P(x_i|\hat{d}_i) = \frac{P(\hat{d}_i|x_i) \cdot P(x_i|x_{\partial i})}{P(\hat{d}_i)} \tag{3.2}$$

$$P(\hat{d}_i) = \int_S P(\hat{d}_i|x_i) \cdot \mathrm{d}P(x_i|x_{\partial i}) \tag{3.3}$$

Still the main problem is to determine a good approximation for $P(x_i|x_{\partial i})$.

3.7 The ICM algorithm

This work has concentrated on the use of the ICM algorithm (iterated conditional modes) to obtain a better approximation to the true scene from our recording which includes a lot of added noise. This algorithm was developed in the early 80's. Some of the idea behind the algorithm was to solve two problems which other procedures used in the restoration of noisy images suffer from: The need for vast amounts of computing power and the problem of producing false correlations over long distances while restoring a picture. The latter problem is reduced because only a few iterations is needed due to the rapid convergence of this algorithm.

3.7.1 The two basic assumptions

The *first assumption* is that the observed intensity or colour of the picture elements are conditionally independent. If S is the entire set of picture elements and x_i the true value for element i, then

$$f(\hat{\mathbf{d}};\mathbf{x}) = \prod_{i \in S} f(\hat{d}_i|x_i) \tag{3.4}$$

This just means that our observed test statistic $\hat{d}$ depends on the state of the single picture element i and is independent of the state of all other picture elements when x_i is given. The function $f(\hat{d}_i|x_i)$ gives the distribution of the noise. If it is not known it often can be estimated from the observed picture.

The *second assumption* is that the probability distribution of the intensity or colour of a picture element without any observation, the prior probability, can be calculated from the state of its defined neighbouhood ∂i. The true picture is a realization of a locally dependent Markov random field $\{p(\mathbf{x})\}$.

3.8 Basic theory for ICM

The aim while doing image restoration is in the majority of cases to maximize $P(\mathbf{x}|\hat{\mathbf{d}})$, the maximum aposteriori probability (MAP) estimator given by expression (3.1). This is most often an overwhelmingly complex task both because it is difficult to obtain expressions for the probabilities in the equation and because it would require vast amounts of computing power.

We state the simplifications of the algorithm and give some equations to gain some insight into how the ICM algorithm performs.

It is an easier task to calculate the MAP for a smaller part of the image than for the whole scene. Trying maximizing the probability of picture element i and its defined neighbourhood given the observed scene, the MAP estimator obtained is

$$P(x_i, x_{\partial i}|\hat{\mathbf{d}}) = P(x_i|x_{\partial i}, \hat{\mathbf{d}}) \cdot P(x_{\partial i}|\hat{\mathbf{d}})$$

This greatly simplifies the task of estimating $\mathbf{x}$. Maximizing with respect to x_i reduces to maximizing

$$P(x_i|x_{\partial i}, \hat{\mathbf{d}}) = \frac{P(\hat{\mathbf{d}}, x_{\partial i}|x_i) \cdot P(x_i)}{P(x_{\partial i}, \hat{\mathbf{d}})} = \frac{P(\hat{\mathbf{d}}|x_{\partial i}, x_i) \cdot P(x_{\partial i}|x_i) \cdot P(x_i)}{P(x_{\partial i}, \hat{\mathbf{d}})} \tag{3.5}$$

$$= \frac{P(\hat{\mathbf{d}}|x_{\partial i}, x_i) \cdot P(x_i|x_{\partial i}) \cdot P(x_{\partial i})}{P(x_{\partial i}, \hat{\mathbf{d}})} \tag{3.6}$$

$$= \frac{P(\hat{d}_i|x_i) \cdot P(x_i|x_{\partial i}) \cdot P(\hat{\mathbf{d}}_{S\setminus i})|x_{\partial i}) \cdot P(x_{\partial i})}{P(x_{\partial i}, \hat{\mathbf{d}})} \tag{3.7}$$

The first three equalities are given by the Bayesian theorem and formulas for conditional probability and the last equality is the first assumption for the algorithm given by (3.4). Hence we have concerning the maximization with respect to x_i:

$$P(x_i, x_{\partial i}|\hat{\mathbf{d}}) \propto P(\hat{d}_i|x_i) \cdot P(x_i|x_{\partial i})$$

That $P(x_i|x_{\partial i})$ is given is the second assumption for the algorithm. Here $P(x_i|x_{\partial i}) = P(x_i|\mathbf{x})$, i.e. the Markov random field is locally dependent.

To summarize the ICM algorithm: The expression that is maximized is

$$P(x_i|x_{\partial i}, \hat{\mathbf{d}}) \propto P(\hat{d}_i|x_i) \cdot P(x_i|x_{\partial i})$$

This is equivalent to maximizing $P(x_i, x_{\partial i}|\hat{\mathbf{d}})$ which is chosen as a substitute for maximizing $P(\mathbf{x}|\hat{\mathbf{d}})$ and is also equivalent to maximizing

$$P(x_i|x_{\partial i}, \hat{d}_i) = \frac{P(\hat{d}_i|x_i, x_{\partial i}) \cdot P(x_i|x_{\partial i})}{P(\hat{d}_i|x_{\partial i})} = \frac{P(\hat{d}_i|x_i) \cdot P(x_i|x_{\partial i})}{P(\hat{d}_i)}$$

The first equality is just the Bayesian theorem with all probabilities conditioned on $x_{\partial i}$ and the second equality follows from the assumption of conditional independence for ICM and $P(\hat{d}_i|\mathbf{x}_{S\setminus i}) = P(\hat{d}_i)$.

In most applications $P(\hat{d}_i|x_i)$ has a normal distribution and a Gibbs distribution is defined for $P(x_i|x_{\partial i})$.

3.8.1 Brief description of how the algorithm is carried out

The ICM is an iterative algorithm. During each iteration the *observed picture* and the *present estimate of the true picture* are combined to get a better estimate of the latter. This is done by just running through the entire set of picture elements, setting the pixels individually to their most likely state calculated from its observed value $\hat{d}_i$ and the present estimate of its neighbourhood, $x_{\partial i}$. Initially the estimate must be set to some value. A common choice is to set it equal to the "maximum likelihood classifier", i.e. the most likely value when neighbours are not taken into account. The algorithm ends after a predetermined number of iterations.

The picture elements in the present estimate of the true picture can either be updated simultaneously (after they have been updated individually and stored in a buffer) or each picture element can be updated immediately after calculation of its most likely value based on its observed value $\hat{d}_i$ and the present estimate of $\mathbf{x}$.

If the latter scheme of updating is used, convergence is guaranteed because

$$P(\mathbf{x}|\hat{\mathbf{d}}) = P(x_i|\mathbf{x}_{S\setminus i}, \hat{\mathbf{d}}) \cdot P(\mathbf{x}_{S\setminus i}) = P(x_i|x_{\partial i}, \hat{\mathbf{d}}) \cdot P(\mathbf{x}_{S\setminus i})$$

$P(\hat{\mathbf{x}}|\hat{\mathbf{d}})$ never decreases during the update of any picture element of $\hat{\mathbf{x}}$ and convergence is assured. In the case of simultaneous updating of the x_i's convergence is not guaranteed. Instead some minor oscillations of $\hat{\mathbf{x}}$ most often occur.

3.9 Approximations for $P(x_i|x_{\partial i})$

As previously mentioned the main problem when using Bayesian restoration is to obtain a simple expression for a good approximation to the prior probability of the estimated true scene. Here two different approaches have been used: It is common to define a potential function which has its minimum value if the picture elements have equal values. Then the prior distribution is approximated by using this potential as the argument to a Gibbs distribution. A common choice for the potential function is The Geman McClure potential. This choice of potential has shown to give good results. A useful property of this potential function in many applications is that it allows for abrupt changes in the values of nearby picture elements but smooths out small (often random) variations.

Secondly an alternative approach for the separation of pictures into *two* main parts, easier to implement and less computing intensive, is described. This choice also has the advantage that estimation of the parameters are easy if a true picture or an approximation to a true picture is given.

Both concepts are based on the general observation that nearby picture elements tend to have equal values.

3.9.1 Using the Geman McClure potential

The use of some potential function to express the prior probability $P(x_i|x_{\partial i})$ is a common implementation of the fact that nearby picture elements most likely have equal values. The potential function is defined such that it has a minimum for equal values of the neighbours. If $P(x_i|x_{\partial i})$ is defined as a monotone function of the potential and otherwise satisfies the requirements for being a probability, i.e. having positive values only and integrating to 1 on the entire real line, it can be used as a probability density function for x_i while applying an iterative algorithm for Bayesian restoration of pictures.

If θ is a parameter vector, a potential function of the state of x_i and its

neighbourhood $\{x_j\}$ is given as

$$V(x_i;\theta) = \sum_{x_j \sim x_i} V_j(x_j, x_i;\theta)$$

$x_j \sim x_i$ means that x_j belongs to the set defined to be the neighbours of x_i. The probability $P(x_i|x_{\partial i})$ is define as a function of the potential, usually a Gibbs distribution:

$$P(x_i|x_{\partial i}) = \frac{1}{Z(\theta)}\, e^{-V(x_i;\theta)}$$

The normalizing constant $Z(\theta)$ is usually not known. Furthermore it depends on the choice of the parameters for the potential function. The former is not a problem while using Bayesian restoration. The latter is a problem in some contexts. We recall that the expression to be maximized during Bayesian restoration is

$$P(x_i|x_{\partial i}, \hat{d}_i) = \frac{f(\hat{d}_i|x_i) \cdot P(x_i|x_{\partial i})}{f(\hat{d}_i)} = \frac{f(\hat{d}_i|x_i) \cdot \frac{1}{Z(\theta)} e^{-V(x_i;\theta)}}{f(\hat{d}_i)} \propto f(\hat{d}_i|x_i) \cdot e^{-V(x_i;\theta)}$$

So $P(x_i|x_{\partial i}, \hat{d}_i)$ is proportional in x_i to $f(\hat{d}_i|x_i) \cdot e^{-V(x_i;\theta)}$. Neither $f(\hat{d}_i) = \int_S f(\hat{d}_i|x_i)\mathrm{d}P(x_i|x_{\partial i})$ nor $\frac{1}{Z(\theta)}$ depends on the state of x_i or its neighbourhood, so in general Bayesian restoration reduces to maximizing the product

$$f(\hat{d}_i|x_i) \cdot e^{-V(x_i;\theta)} \tag{3.8}$$

with respect to x_i for each i.

If $\hat{d}_i|x_i \sim N(x_i, \sigma^2)$ we obtain a simpler expression to maximize:

$$f(\hat{d}_i|x_i) \cdot e^{-V(x_i;\theta)} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}(\frac{\hat{d}_i - x_i}{\sigma})^2} \cdot e^{-V(x_i;\theta)} \propto e^{-(\frac{1}{2}(\frac{\hat{d}_i - x_i}{\sigma})^2 + V(x_i;\theta))}$$

Maximizing $e^{-(\frac{1}{2}(\frac{\hat{d}_i - x_i}{\sigma})^2 + V(x_i;\theta))}$ is equivalent to minimizing $\frac{1}{2}(\frac{\hat{d}_i - x_i}{\sigma})^2 + V(x_i;\theta) = \frac{1}{2}(\frac{\hat{d}_i - x_i}{\sigma})^2 + \sum_{x_j \sim x_i} V_j(x_j, x_i;\theta)$. So maximizing $P(x_i|\hat{d}_i)$ when using a Gibbs distribution with a potential function to express the prior distribution of x_i and Gaussian noise is done by minimizing

$$\frac{1}{2}(\frac{\hat{d}_i - x_i}{\sigma})^2 + \sum_{x_j \sim x_i} V_j(x_j, x_i;\theta)$$

The Geman McClure potential is given by

$$V(x_j, x_i;\beta,\delta) = -\frac{\beta}{1 + (\frac{x_i - x_j}{\delta})^2} \tag{3.9}$$

σ is still the variance of the noise of $\hat{d}_i$ and is often known or can be estimated from the picture. But the estimation of the parameters β and δ is a complicated task because the normalizing constant $Z(\beta,\delta)$ depends on the parameters to

be estimated and expressing the constant as a function of these parameters is difficult. If the relationship between the parameters in the potential and the normalizing constant could be easily given, β and δ could be estimated by maximizing the likelihood of a given picture. This potential is an artificial construct but it has proved to give good result when used during Bayesian picture restoration.

3.9.2 Maximizing $f(\hat{d}_i|x_i) \cdot P(x_i|x_{\partial i})$ by using the Bayesian version of the Neyman-Pearson lemma

Our aim is to separate *two* populations: Background and object. The Neyman-Pearson lemma stated on page 4 tells how to choose the threshold for sorting a given picture element into either of two populations and the threshold by taking prior probability into account is given by (2.6). Because we have only two populations and hence only two values for x_i, $f(\hat{d}_i|x_i)$ is either the probability density for $\hat{d}_i$ given picture element i shows the background or the probability density for $\hat{d}_i$ given picture element i shows the object. $P(x_i|x_{\partial i})$ is the prior probability for picture element i taking the value x_i given the states of its neighbours. Since x_i is restricted to 2 values, $P(x_i|x_{\partial i})$ can take either of two values summing to 1. During the maximizing of
$f(\hat{d}_i|x_i) \cdot P(x_i|x_{\partial i})$ the value for x_i giving the greater product is chosen. Maximizing $f(\hat{d}_i|x_i) \cdot P(x_i|x_{\partial i})$ corresponds to classifying by choosing the threshold according to (2.6) setting $p = P(x_i = \mu_0|x_{\partial i})$. So $P(x_i|x_{\partial i}, \hat{d}_i)$ can be maximized according to the Bayesian theorem by selecting the threshold as given by (2.6) with $p = P(x_i = \mu_0|x_{\partial i})$ and classifying voxel i by using this threshold for $\hat{d}_i$.

This method also have a nice property compared to using a potential and a Gibbs distribution: There is no unknown normalization constant, so the maximized probability is an estimate of the reliability of the estimate of the state of voxel i. Hence the uncertainty of the estimated state of the voxels can be included in the restored picture, making it possible to define "regions of doubt". Furthermore it is an attractive feature of this method that the link to basic theory is more directly. The function that expresses the prior probability as a function of the state of the elements defined as the neighbourhood does not need the construction of a potential function used as an argument to a Gibbs distribution.

Perhaps the greatest advantage compared to using a Gibbs distribution with a chosen potential function is that it is much simpler to estimate the parameters of the proposed approximation for $P(x_i = \mu_0|x_{\partial i})$. As mentioned before the problem of estimating the parameters of the potential function is that the normalizing constant for the distribution is not only unknown, but it depends upon the parameters for the potential function. Hence the common maximization of the likelihood cannot be used to obtain the values of the parameters.

Figure 3.4 clarifies the use of the thresholds given by the "Bayesian version" of the Neyman-Pearson lemma while running ICM for classification into either

of two populations. The expression to maximize with respect to x_i is

$$P(x_i|x_{\partial i}, \hat{d}_i) = \frac{f(\hat{d}_i|x_i) \cdot P(x_i|x_{\partial i})}{f(\hat{d}_i)} \tag{3.10}$$

$f(\hat{d}_i)$ is not the mixture distribution estimated from our recorded data, but the "total probability", the distribution given by expression (3.3), in this case a mixture distribution with the same normal distributions as our observed mixture distribution, but the parameter p in the Bernoulli part of the hierarchical distribution is substituted by $P(x_i = \mu_0|x_{\partial i})$ which is a function of the elements in the neighbourhood of i:

$$f(\hat{d}_i) = f(\hat{d}_i|x_i = \mu_0)P(x_i = \mu_0|x_{\partial i}) + f(\hat{d}_i|x_i = \mu_1)P(x_i = \mu_1|x_{\partial i})$$

The red curves are two plots of these mixture distribution for different values of $P(x_i = \mu_0|x_{\partial i})$ scaled to fit better into the figure, i.e. they are scaled plots of the denominator in (3.10). The blue curves with discontinuous derivatives at two points are the corresponding plots of
$max_{x_i}(f(\hat{d}_i|x_i = 0)P(x_i = 0|x_{\partial i}), f(\hat{d}_i|x_i = 1)P(x_i = 1|x_{\partial i}))$, scaled wtih the same factor as the red curves, i.e. they are scaled plots of the numerator of (3.10) after maximizing with respect to either of the two possible values for x_i. Finally the black curves are the corresponding maximized expression (3.10). It estimates the probability of picture i being correctly classified. The thresholds for classifying either into the population of the background or into the population of the object are given by the discontinuities of the derivatives of the blue and black curves.

3.9.3 Estimating $P(x_i|x_{\partial i})$ by counting neighbours in either state

It remains to estimate $P(x_i|x_{\partial i})$ as a function of the values of the elements in the neighbourhood. x_i can take only the two values μ_0 and μ_1. Because
$P(x_i = \mu_1|x_{\partial i})$ = 1 - $P(x_i = \mu_0|x_{\partial i})$ it suffices to obtain a function for one of these two probabilities. $P(x_i = \mu_0|x_{\partial i})$ is our choice since it coincides with p in our expression for the mixture distribution.

The common approach for estimating parameters of probability functions is using maximum likelihood. The values of the parameters making the whole picture most likely is chosen as the parameters. Another approach is also tried here: Finding the parameter values which is gives fewest changes to the scene. The likelihood function to be used for estimating the parameter vector θ for $P(x_i = \mu_0|x_{\partial i})$ is

$$L(\theta; \mathbf{x}) = f(\mathbf{x}; \theta) = \prod_{i \in S} P(x_i|x_{\partial i}) \tag{3.11}$$

We have either $x_i = \mu_0$ or $x_i = \mu_1$, so the likelihood function can be written

$$L(\theta; \mathbf{x}) = \prod_{i \in S} (\mathbf{I}(x_i = \mu_0)P(x_i = 0|x_{\partial i}) + \mathbf{I}(x_i = \mu_1)P(x_i = 1|x_{\partial i})) \tag{3.12}$$

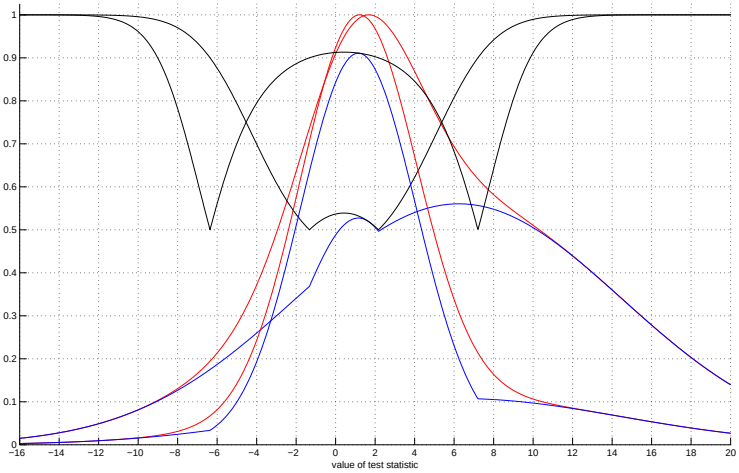

Figure 3.4: Curves showing separation of picture into two populations by maximizing expression (3.2). Maximization with $P(x_i = \mu_0|x_{\partial i}) = 0.25$ and $P(x_i = \mu_0|x_{\partial i}) = 0.75$ shown. The density curves are normalized such that the maximum of the mixture distributions equals 1. The red curve is the mixture distributions with p equal to 0.25 and 0.75. The blue curves is the maximum of the numerator in (3.2) with $P(x_i = \mu_0|x_{\partial i}) = 0.25$ and $P(x_i = \mu_0|x_{\partial i}) = 0.75$. The black curves is the value of the entire expression maximized.

As is commonly the case when numerically calculating the likelihood with a large number of elements in S the likelihood function above cannot be used due to numerical underflow in the computer. But the log likelihood function can be used to cope with this problem:

$$log(L(\theta; \mathbf{x})) = \sum_{i \in S} log(P(x_i | x_{\partial i})) \tag{3.13}$$

So the function to maximize with respect to θ can be evaluated by running through the entire set of elements in the picture, calculating the prior probability $P(x_i = \mu_0 | x_{\partial i})$, setting the corresponding term in the sum equal to $log(P(x_i = \mu_0 | x_{\partial i}))$ if the element shows background, otherwise the corresponding term is set equal to $log(1 - P(x_i = \mu_0 | x_{\partial i}))$.

Chapter 4

Testing the robustness of the separation

4.1 The observed information matrix

Observed information is given as the second derivative of the minus log likelihood function:

$$I_0(\theta) = -\frac{\mathrm{d}^2 log(L(\hat{d}|\theta))}{\mathrm{d}\theta^2} \tag{4.1}$$

For large sample sizes, which is usually the case when estimating from all the elements is a picture, the maximum likelihood estimator will have a distribution close to normal:

$$\hat{\theta} \sim N(\hat{\theta}, \frac{1}{I_0(\theta)})$$

If several parameters are to be estimated, an observed information matrix can be calculated. The information matrix with n parameters given by the n-dimensional parameter vector $\theta = [\theta_1 \dots \theta_n]$ is

$$I_0(\theta) = \begin{bmatrix} -\frac{\partial^2 log(L(\theta;\hat{\mathbf{d}}))}{\partial\theta_1^2} & \cdots & -\frac{\partial^2 log(L(\theta;\hat{\mathbf{d}}))}{\partial\theta_1\partial\theta_n} \\ \vdots & \ddots & \vdots \\ -\frac{\partial^2 log(L(\theta;\hat{\mathbf{d}}))}{\partial\theta_n\partial\theta_1} & \cdots & -\frac{\partial^2 log(L(\theta;\hat{\mathbf{d}}))}{\partial\theta_n^2} \end{bmatrix}$$

Analogous to the single-dimensional case the parameters estimated from various pictures of identical scenes will have an asymptotic multinormal distribution with the inverted observed information matrix as the covariance matrix:

$$\hat{\theta} \sim N(\hat{\theta}, I_0^{-1}(\theta)) \tag{4.2}$$

This can be used for simulating a set of parameters from estimations and testing separation using the parameters in this set. This will give an indication of the robustness of the separation of the observed image.

4.2 The information matrix for the normal-Bernoulli density

In our special case we have 5 parameters: $\theta = [\theta_1\theta_2\theta_3\theta_4\theta_5]' = [p\ \mu_0\ \mu_1\ \sigma_0\ \sigma_1]'$. $L(\theta;\mathbf{y}) = L(p,\mu_0,\mu_1,\sigma_0,\sigma_1;\hat{\mathbf{d}}) = f(\hat{\mathbf{d}};p,\mu_0,\mu_1,\sigma_0,\sigma_1)$. We have

$$f(\hat{\mathbf{d}};\sigma_1,\mu_0,\mu_1,\sigma_0,\sigma_1) = \prod_{i=1}^{n}(\frac{p}{\sqrt{2\pi\sigma_0^2}}e^{-\frac{(\hat{d}_i-\mu_0)^2}{2\sigma_0^2}} + \frac{1-p}{\sqrt{2\pi\sigma_1^2}}e^{-\frac{(\hat{d}_i-\mu_1)^2}{2\sigma_1^2}}) \quad (4.3)$$

This gives the information matrix

$$I_0(\theta) = \begin{bmatrix} -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial p^2} & -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial p\partial\mu_0} & -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial p\partial\mu_1} & -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial p\partial\sigma_0} & -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial p\partial\sigma_1} \\ -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\mu_0\partial p} & -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\mu_0^2} & -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\mu_0\partial\mu_1} & -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\mu_0\partial\sigma_0} & -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\mu_0\partial\sigma_1} \\ -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\mu_1\partial p} & -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\mu_1\partial\mu_0} & -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\mu_1^2} & -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\mu_1\partial\sigma_0} & -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\mu_1\partial\sigma_1} \\ -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\sigma_0\partial p} & -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\sigma_0\partial\mu_0} & -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\sigma_0\partial\mu_1} & -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\sigma_0^2} & -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\sigma_0\partial\sigma_1} \\ -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\sigma_1\partial p} & -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\sigma_1\partial\mu_0} & -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\sigma_1\partial\mu_1} & -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\sigma_1\partial\sigma_0} & -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\sigma_1^2} \end{bmatrix}$$

The matrix is symmetric:

$$-\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\theta_m\partial\theta_n} = -\frac{\partial^2 logf(\hat{\mathbf{d}};\theta)}{\partial\theta_n\partial\theta_m} \quad (4.4)$$

4.3 Deriving expressions for the entries in the observed information matrix

4.3.1 Useful differentiation rules for the normal distribution

In our case we have a normal mixture distribution. The partial differentiations mostly involves taking the derivative of the normal pdf with respect to the parameters μ and σ. Stating a couple of rules for doing this will save a lot of work.

The normal distribution has good properties for analytical treatment. Differentiation with respect to the parameters μ and σ gives relatively simple expressions. These results will be used extensively during the derivation of the

information matrix and may be easily verified by using standard differentiation rules. Partial differentiation with respect to μ gives

$$\frac{\partial}{\partial\mu}(\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2}(\frac{y-\mu}{\sigma})^2}) = (\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2}(\frac{y-\mu}{\sigma})^2}) \cdot \frac{1}{\sigma}(\frac{y-\mu}{\sigma}) \tag{4.5}$$

So the partial derivative of the normal distribution with respect to its mean value is obtained by just multiplying the density function by the inverse of its standard deviation and its "standardized" argument, $\frac{y-\mu}{\sigma}$.

The formula for the partial derivative with respect to σ is a bit more complicated but still relatively simple. Partial differentiation gives

$$\frac{\partial}{\partial\sigma}(\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2}(\frac{y-\mu}{\sigma})^2}) = (\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2}(\frac{y-\mu}{\sigma})^2}) \cdot \frac{1}{\sigma}((\frac{y-\mu}{\sigma})^2 - 1)) \tag{4.6}$$

This expression is also easy to remember: The normal density function is multiplied by the inverse of its standard deviation and the square of its "standardized" argument minus one.

4.3.2 Calculating the entries in the information matrix

The information matrix is symmetric, so it suffices to calculate either the upper or the lower triangular matrix. For a 5x5 matrix it is necessary to calculate 15 values and the rest is given by the symmetry. The matrix is calculated from

$$logf(\hat{\mathbf{d}};\theta) = log\prod_{i=1}^{n}(\frac{p}{\sqrt{2\pi\sigma_0^2}}e^{-\frac{(\hat{d}_i-\mu_0)^2}{2\sigma_0^2}} + \frac{1-p}{\sqrt{2\pi\sigma_1^2}}e^{-\frac{(\hat{d}_i-\mu_1)^2}{2\sigma_1^2}}) \tag{4.7}$$

$$= \sum_{i=1}^{n} log(\frac{p}{\sqrt{2\pi\sigma_0^2}}e^{-\frac{(\hat{d}_i-\mu_0)^2}{2\sigma_0^2}} + \frac{1-p}{\sqrt{2\pi\sigma_1^2}}e^{-\frac{(\hat{d}_i-\mu_1)^2}{2\sigma_1^2}}) \tag{4.8}$$

Each element is a sum of n similar terms, each term is calculated for a particular $\hat{d}_i$, i.e. the information matrix is a sum of n matrices, each matrix in the sum having similar entries which depends on one particular calculated value of our test statistic $\hat{d}_i$.

We proceed by deriving the expressions for the entries in the matrices, each to be calculated for one of the values of $\hat{d}_i$ and eventually summed to give the information matrix. For clarity and brevity we use the following in the expressions for the matrix entries

$$f_0 = f_0(\hat{d}_i;\mu_0,\sigma_0) = f(\hat{d}_i;\mu_0,\sigma_0|H_0) = \frac{1}{\sqrt{2\pi}\sigma_0}e^{-\frac{1}{2}(\frac{\hat{d}_i-\mu_0}{\sigma_0})^2} \tag{4.9}$$

$$f_1 = f_1(\hat{d}_i;\mu_1,\sigma_1) = f(\hat{d}_i;\mu_1,\sigma_1|H_1) = \frac{1}{\sqrt{2\pi}\sigma_1}e^{-\frac{1}{2}(\frac{\hat{d}_i-\mu_1}{\sigma_1})^2} \tag{4.10}$$

$$f = f(\hat{d}_i;\theta) = \frac{p}{\sqrt{2\pi}\sigma_0}e^{-\frac{1}{2}(\frac{\hat{d}_i-\mu_0}{\sigma_0})^2} + \frac{1-p}{\sqrt{2\pi}\sigma_1}e^{-\frac{1}{2}(\frac{\hat{d}_i-\mu_1}{\sigma_1})^2} \tag{4.11}$$

The entries $I_{mn,\,i}$ in row m, column n in matrix i in the sum are given by

$$I_{mn,\,i}(\theta,\hat{d}_i) = -\frac{\partial^2 log f(\hat{d}_i;\theta)}{\partial\theta_m\partial\theta_n}, \qquad \theta = [\theta_1\ \theta_2\ \theta_3\ \theta_4\ \theta_5]' = [p\ \mu_0\ \mu_1\ \sigma_0\ \sigma_1]' \tag{4.12}$$

Because of the symmetry,

$$I_{mn,\,i}(\theta,\hat{d}_i) = I_{nm,\,i}(\theta,\hat{d}_i) \tag{4.13}$$

we only derive the expressions from the diagonal entries and downwards, using the same expressions for the upper half of the matrix. We are now ready to find the partial derivatives.

$$\frac{\partial}{\partial p} log f = \frac{f_0 - f_1}{f} \tag{4.14}$$

Proceeding by a second partial derivation with respect to each of the parameters in turn we obtain the first column of matrix i in the sum of matrices:

$$-\frac{\partial^2}{\partial p^2} log f = (\frac{f_0 - f_1}{f})^2$$

$$-\frac{\partial^2}{\partial\mu_0\partial p} log f = \frac{(p(f_0 - f_1) - f)f_0}{f^2} \cdot \frac{\hat{d}_i - \mu_0}{\sigma_0^2}$$

$$-\frac{\partial^2}{\partial\mu_1\partial p} log f = \frac{((1-p)(f_0 - f_1) + f)f_1}{f^2} \cdot \frac{\hat{d}_i - \mu_1}{\sigma_1^2}$$

$$-\frac{\partial^2}{\partial\sigma_0\partial p} log f = \frac{(p(f_0 - f_1) - f)f_0}{f^2} \cdot (\frac{(\hat{d}_i - \mu_0)^2}{\sigma_0^3} - \frac{1}{\sigma_0})$$

$$-\frac{\partial^2}{\partial\sigma_1\partial p} log f = \frac{((1-p)(f_0 - f_1) + f)f_1}{f^2} \cdot (\frac{(\hat{d}_i - \mu_1)^2}{\sigma_1^3} - \frac{1}{\sigma_1})$$

We now have all the entries in the first column and also in the first row due to the symmetry. We proceed to find the remaining entries in the second column.

$$\frac{\partial}{\partial\mu_0} log f = \frac{pf_0}{f} \cdot \frac{\hat{d}_i - \mu_0}{\sigma_0^2}$$

Next we obtain the entries in the second column from the diagonal entry and below:

$$-\frac{\partial^2}{\partial\mu_0^2} log f = \frac{(pf_0 - f)pf_0}{f^2} \cdot (\frac{\hat{d}_i - \mu_0}{\sigma_0^2})^2 \quad + \quad \frac{pf_0}{\sigma_0^2 f}$$

$$-\frac{\partial^2}{\partial\mu_1\partial\mu_0} log f = \frac{pf_0(1-p)f_1}{f^2} \cdot \frac{\hat{d}_i - \mu_0}{\sigma_0^2} \cdot \frac{\hat{d}_i - \mu_1}{\sigma_1^2}$$

$$-\frac{\partial^2}{\partial\sigma_0\partial\mu_0} log f = (\frac{(pf_0 - f)pf_0}{f^2}(\frac{(\hat{d}_i - \mu_0)^2}{\sigma_0^3} - \frac{1}{\sigma_0}) + \frac{2pf_0}{\sigma_0 f})\frac{\hat{d}_i - \mu_0}{\sigma_0^2}$$

$$-\frac{\partial^2}{\partial\sigma_1\partial\mu_0}logf = \frac{pf_0(1-p)f_1}{f^2}\cdot\frac{\hat{d}_i-\mu_0}{\sigma_0^2}(\frac{(\hat{d}_i-\mu_1)^2}{\sigma_1^3} - \frac{1}{\sigma_1})$$

We are now finished with the first two columns and the first two rows. To obtain the remaing entries is the third column and the third row, we start with the partial derivative with respedt to μ_1:

$$\frac{\partial}{\partial\mu_1}logf = \frac{(1-p)f_1}{f}\cdot\frac{\hat{d}_i-\mu_1}{\sigma_1^2}$$

$$-\frac{\partial^2}{\partial\mu_1^2}logf = \frac{((1-p)f_1-f)(1-p)f_1}{f^2}\cdot(\frac{\hat{d}_i-\mu_1}{\sigma_1^2})^2 + \frac{(1-p)f_1}{\sigma_1^2 f}$$

$$-\frac{\partial^2}{\partial\sigma_0\partial\mu_1}logf = \frac{pf_0(1-p)f_1}{f^2}\cdot(\frac{(\hat{d}_i-\mu_0)^2}{\sigma_0^3} - \frac{1}{\sigma_0})\cdot\frac{\hat{d}_i-\mu_1}{\sigma_1^2}$$

$$-\frac{\partial^2}{\partial\sigma_1\partial\mu_1}logf = (\frac{((1-p)f_1-f)(1-p)f_1}{f^2}(\frac{(\hat{d}_i-\mu_1)^2}{\sigma_1^3} - \frac{1}{\sigma_1}) + \frac{2(1-p)f_1}{\sigma_1 f})\frac{\hat{d}_i-\mu_1}{\sigma_1^2}$$

The entries of the first three columns and the first three rows are now found. We obtain for the first partial derivative with respect to σ_0:

$$\frac{\partial}{\partial\sigma_0} = \frac{pf_0}{f}\cdot(\frac{(\hat{d}_i-\mu_0)^2}{\sigma_0^3} - \frac{1}{\sigma_0})$$

The next entries to be found is the next last diagonal entry and the entry below:

$$-\frac{\partial^2}{\partial\sigma_0^2}logf = \frac{(pf_0-f)pf_0}{f^2}\cdot(\frac{(\hat{d}_i-\mu_0)^2}{\sigma_0^3} - \frac{1}{\sigma_0})^2 + \frac{pf_0}{f}\cdot(3(\frac{\hat{d}_i-\mu_0}{\sigma_0^2})^2 - \frac{1}{\sigma_0^2})$$

$$-\frac{\partial^2}{\partial\sigma_1\partial\sigma_0}logf = \frac{pf_0(1-p)f_1}{f^2}(\frac{(\hat{d}_i-\mu_0)^2}{\sigma_0^3} - \frac{1}{\sigma_0})(\frac{(\hat{d}_i-\mu_1)^2}{\sigma_1^3} - \frac{1}{\sigma_1})$$

Now the only remaining entry is the lower right diagonal element which can be found from the expression for the next last diagonal element by substituting $1-p$ for p, f_1 for f_0, μ_1 for μ_0 and σ_1 for σ_0:

$$-\frac{\partial^2}{\partial\sigma_1^2}logf = \frac{((1-p)f_1-f)(1-p)f_1}{f^2}(\frac{(\hat{d}_i-\mu_1)^2}{\sigma_1^3} - \frac{1}{\sigma_1})^2 + \frac{(1-p)f_1}{f}(3(\frac{\hat{d}_i-\mu_1}{\sigma_1^2})^2 - \frac{1}{\sigma_1^2})$$

$\frac{\hat{d}_i-\mu_k}{\sigma_k^2}$ and $\frac{(\hat{d}_i-\mu_k)^2}{\sigma_k^3} - \frac{1}{\sigma_k}$, $k=1,2$ occur frequently in the matrix entries, so the calculation of matrix terms in the sum can easily be coded by substitutions for these expressions. The values for these two expressions, f, f_0 and f_1 must be calculated for the corresponding value of $\hat{d}_i$ for the term in the matrix sum.

Chapter 5

Simple probability model for fMRI recordings

5.1 Defining our simple model

The main task of fMRI is to distinguish between activated and not activated parts of the brain while the person being tested is exposed to some kind of stimuli. As an elementary introduction to the problem a simple probability model is proposed. To simplify matters in the outset, we make a few assumptions:

- The signal from a voxel[1] has either of two levels corresponding to activated or not activated state
- Noise has a normal distribution
- Noise is equally spread throughout the entire volume

The estimate of the mean d of the difference between the average of the signal during the times a voxel may be activated by the applied stimuli and the average strength of the signal when the stimuli is not applied is chosen.

Study of fMRI recordings shows that the assumption of gaussian noise is met quite well as expected due to the Central Limit Theorem. But the strength of the noise varies a lot throughout the volume. The assumption of a signal taking either of two levels does not hold due to properties of the brain matter. Nevertheless a study of this model is a good introduction which also applies to similar detection and separation problems.

5.2 Format of fMRI recordings

The fMRI recording equipment measures the the oxygen content in the blood, utilizing the BOLD[2] response to indirectly image active areas of the brain. Ac-

[1] voxel: volume element, cf. pixel: picture element
[2] BOLD: blood oxygenation level dependent

tive areas of the brain exibits increased levels of oxygen. 3-dimensional pictures of the oxygen level in the various parts of the brain are recorded. Typically the entire data set comprises $96 \cdot 96 \cdot 29 = 267,264$ voxels. Of these only about 20,000 image grey brain matter having the capability to be in an activated state. The brain is scanned once every third second while the person tested is alternating between rest and some kind of stimuli or activity. The periods of rest and activity are repeated once a minute. There is no standard length of the recording, but usually the entire recording contains data for about 5 minutes. Thus a time series of about 100 values is obtained for each voxel.

Figure 5.1 shows the recorded time series for one of the voxels. In this particular case the fluctuations of the signal due to alternating periods of rest and activity can be anticipated. Half a minute of rest is followed by half a minute of activity. This is repeated during the time of recording. The time series shown is from the part of the brain most clearly being activated by the stimuli. Unfortunately it is usually much harder to see from the time series whether the voxel is in its activated state.

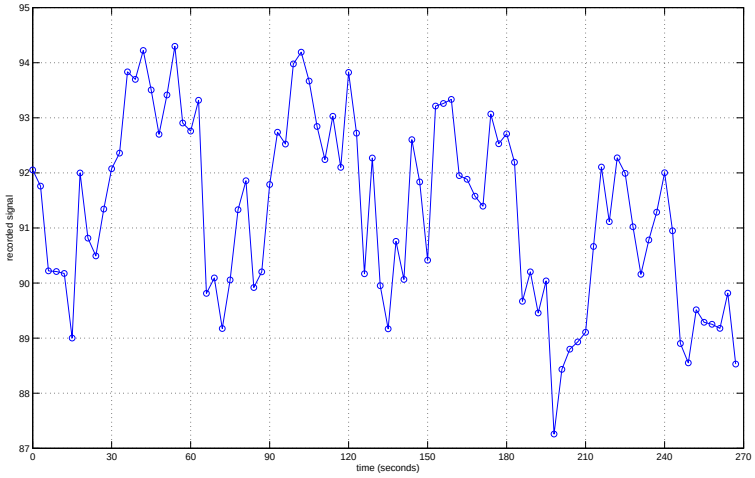

Figure 5.1: Time series of signal from activated voxel.

5.3 Calculating our test statistic and the noise level

After taking a look at figure 5.2 we decide for using the first 2 and the last 9 samples recorded during each cycle of rest and activity to calculate the signal level during the time of activity. The remaining 9 samples, 3 to 11, is used for calculating the signal level during the time of rest.

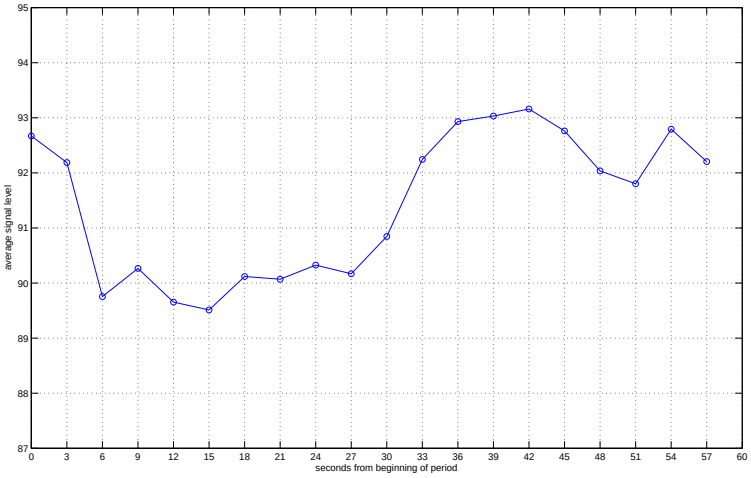

Figure 5.2: Signal levels obtained by averaging the 4 1/2 periods each of length one minute in figure 5.1.

We obtain the value for the signal level during time of activity for a particular voxel by calculating the average in the time series of all the samples assumed to display the signal level during time of activity. The signal level during rest is obtained similarly. The difference between these averages is used as test statistic.

The calculations of means and variances, in particular computer programs, is made more complicated by the fact that the length of the recordings is not a whole number of cycles of the stimuli, neither of fixed length. We could of course truncate the signal to a fixed length to simplify matters. But we prefer using all the available samples for not reducing the picture quality.

For this recording of 90 samples, the measured samples in the time series being $y_i,\ i = 1, 2, \ldots 90$, the average of the signal during the period of rest is

$$\bar{y}_{rest} = \frac{1}{44}(\sum_{j=0}^{4}\sum_{i=3}^{10} y_{i+20j} + \sum_{j=0}^{3} y_{11+20j}) \tag{5.1}$$

Similarly the average of the signal during the period of activity is

$$\bar{y}_{activity} = \frac{1}{46}(\sum_{j=0}^{4}\sum_{i=1}^{2} y_{i+20j} + \sum_{j=0}^{3}\sum_{i=12}^{20} y_{i+20j}) \tag{5.2}$$

Hence we obtain the value of our test statistic:

$$\begin{aligned}\hat{d} &= \bar{y}_{activity} - \bar{y}_{rest} \qquad (5.3)\\ &= \frac{1}{46}(\sum_{j=0}^{4}\sum_{i=1}^{2} y_{i+20j} + \sum_{j=0}^{3}\sum_{i=12}^{20} y_{i+20j}) - \frac{1}{44}(\sum_{j=0}^{4}\sum_{i=3}^{10} y_{i+20j} + \sum_{j=0}^{3} y_{11+20j})\end{aligned}$$

Assuming independence the variance of the test statistic is given by

$$\begin{aligned}Var(\hat{d}) &= Var(\frac{1}{46}(\sum_{j=0}^{4}\sum_{i=1}^{2} y_{i+20j} + \sum_{j=0}^{3}\sum_{i=12}^{20} y_{i+20j})\\ &\quad - \frac{1}{44}(\sum_{j=0}^{4}\sum_{i=3}^{10} y_{i+20j} + \sum_{j=0}^{3} y_{11+20j}))\\ &= \frac{1}{46^2}(\sum_{j=0}^{4}\sum_{i=1}^{2} Var(y_i) + \sum_{j=0}^{3}\sum_{i=12}^{20} Var(y_i))\\ &\quad + \frac{1}{44^2}(\sum_{j=0}^{4}\sum_{i=3}^{10} Var(y_i) + \sum_{j=0}^{3} Var(y_i))\\ &= \frac{1}{46}Var(y_i) + \frac{1}{44}Var(y_i) = \frac{90}{2024}Var(y_i) \qquad (5.4)\end{aligned}$$

The values in the time series can be modeled as

$$y_i = x_i + \varepsilon_N, \qquad i = 1, 2, \ldots 90 \qquad (5.5)$$

x_i is the true value of sample i and $\varepsilon_N \sim N(0, \sigma^2_{series})$ is noise. The fact that the time series is periodic with 20 values in each cycle can be used to estimate the noise of each sample. d_i, $d_i + 20, \ldots$ have equal means and distribution. 20 sample variances, $i = 1, \ldots 20$ can be found, then they can be pooled together to get a more precise estimate of σ^2_{series}.

Sample variances S_i^2, $i = 1, \ldots 10$ are each found from 5 samples by inserting the values y_{i+20j}, $j = 0, \ldots 4$ into the following expression

$$S_i^2 = \frac{1}{4}\sum_{j=0}^{4}(y_{i+20j} - \bar{y}_i)^2 \qquad (5.6)$$

$$\bar{y}_i = \frac{1}{5}\sum_{j=0}^{4} y_{i+20j} \qquad (5.7)$$

$$\frac{4S_i^2}{\sigma^2_{series}} \sim \chi^2_4 \qquad (5.8)$$

Sample variances S_i^2, $i = 11, \ldots 20$ are each calculated from 4 samples

$$S_i^2 = \frac{1}{3}\sum_{j=0}^{3}(y_{i+20j} - \bar{y}_i)^2 \qquad (5.9)$$

$$\bar{y}_i = \frac{1}{4}\sum_{j=0}^{3} y_{i+20j} \tag{5.10}$$

$$\frac{3S_i^2}{\sigma_{series}^2} \sim \chi_3^2 \tag{5.11}$$

Adding all these χ^2 variables gives a χ_{70}^2 variable, so the pooled sample variance is given by

$$\frac{4(S_1^2 + \ldots + S_{10}^2) + 3(S_{11}^2 + \ldots + S_{20}^2)}{\sigma_{series}^2} \sim \chi_{70}^2 \tag{5.12}$$

$$\frac{70S_p^2}{\sigma_{series}^2} \sim \chi_{70}^2 \tag{5.13}$$

As an unbiased estimate for σ_{series}^2 we obtain from the 90 samples

$$S_p^2 = \frac{1}{70}(4(S_1^2 + \ldots + S_{10}^2) + 3(S_{11}^2 + \ldots + S_{20}^2)) \tag{5.14}$$

5.4 Detection capabilities using our simple model

5.4.1 Detection of the time series given in figure 5.1

To give an idea of the capability of detecting activated voxels with our simple probability model we look at the time series in figure 5.1 known to be the time series of an activated voxel. We estimate the mean of our test statistic and the noise of the time series to get some information on the variability of the test statistic.

In this case we have a single time series of a voxel known to be in the activated state. If we use it for estimation of μ and σ the calculated value of $\hat{d}$ is our estimate of μ and the single calculaton of the variability of $\hat{d}$ is our estimate of its variability. We obtain

$$\hat{d} = 2.46$$

To obtain information on the variability of $\hat{d}$, we calculate S_p obtaining

$$S_p^2 = 1.03$$

If this is considered as the value of $Var(y_i)$, we obtain

$$Var(\hat{d}) = \frac{90}{2024}Var(y_i) = \frac{90}{2024}1.03 = 0.046$$

$$\sigma = \sqrt{0.046} = 0.21$$

The preceding calculations give a point estimate of μ and $Var(\hat{d})$. The given estimates imply very good detection capability for time series like the series given in figure 5.1. We now do some calculations on the precision of these

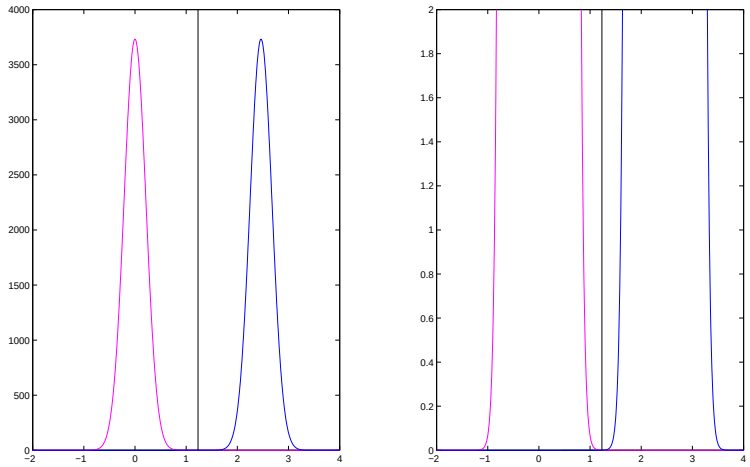

Figure 5.3: Detection capability according to our point estimate of μ and σ. The black vertical line is the threshold given by (2.4). The cyan and blue curves are defined as in figures 3.2 and 3.3. The plot to the right gives a magnification of the y axis.

estimates. We decide for deducing something about the error rate with more than 95% confidence. Regarding the variance we have

$$P(\frac{70S_p^2}{\sigma_{series}^2} > \chi^2_{70,1-\sqrt{0.95}}) = \sqrt{0.95} \tag{5.15}$$

$$P(\sigma_{series}^2 < \frac{70S_p^2}{\chi^2_{70,1-\sqrt{0.95}}}) = \sqrt{0.95} \tag{5.16}$$

$$\sigma^2_{series1} = \frac{70S_p^2}{\chi^2_{70,1-\sqrt{0.95}}} = \frac{70 \cdot 1.03}{48.8} = 1.47 \tag{5.17}$$

$$\sigma_1^2 = \frac{90}{2024}\sigma^2_{series} = \frac{90}{2024}1.47 = 0.065 \tag{5.18}$$

$$\sigma_1 = 0.26 \tag{5.19}$$

Regarding μ we have

$$P(\frac{\hat{d} - \mu}{\sigma} < Z_{\sqrt{0.95}}) = \sqrt{0.95} \tag{5.20}$$

$$P(\mu > \hat{d} - \sigma Z_{\sqrt{0.95}}) = \sqrt{0.95} \tag{5.21}$$

$$\mu_1 = \hat{d} - \sigma_1 Z_{\sqrt{0.95}} = 2.46 - 0.26 \cdot 1.95 = 1.96 \tag{5.22}$$

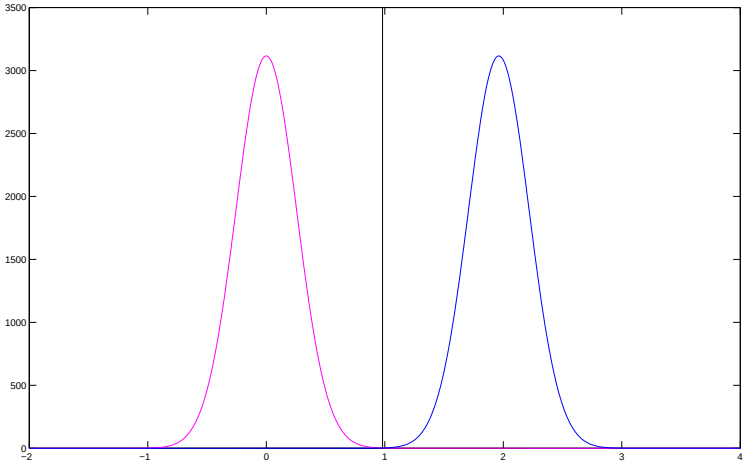

Figure 5.4: Diagram of the case $\sigma = \sigma_1$ and $\mu = \mu_1$. The black vertical line is the threshold given by (2.4). The cyan and blue curves are defined as in figure 5.3. The threshold is chosen in the same way. With probability $\sqrt{0.95}$, $\sigma < \sigma_1$ and with the same probability $\mu > \mu_1$. These values of σ and μ are a subset of the values having equal or better seperation of the two peaks. Estimates of μ and σ are independent random variables (because means and sample variances and functions of them are), so with probability greater than 0.95, detection capability is better than for the values in this diagram.

From (2.7) we obtain for the error rate given $\sigma = \sigma_1$ and $\mu = \mu_1$

$$\begin{aligned} P(error|\sigma_1, \mu_1) &= \int_{\frac{\mu_1}{2}}^{+\infty} \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{x^2}{2\sigma_1^2}} \, \mathrm{d}x \\ &= \int_{\frac{\mu_1}{2\sigma_1}}^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} \, \mathrm{d}z \end{aligned} \tag{5.23}$$

$$\frac{\mu_1}{2\sigma_1} = 3.83$$

So with more than 95% confidence:

$$P(error) \leq 0.000063 \tag{5.24}$$

5.4.2 Calculating variance for a specified error rate

Equation (5.23) can be used to find how much noise a time series can contain if a given error rate is not to be exceeded. If the state of a voxel is to be classified correctly with 95% confidence, we find from statistical tables or program

packages

$$\frac{\mu}{2\sigma} = 1.64$$

Using the estimated value from the time series in figure 5.1 as the value for μ we get

$$\sigma = \frac{2.46}{2 \cdot 1.64} = 0.75$$

By using equation (5.4) the corresponding variance for the samples in the time series is

$$Var(y_i) = \frac{2024\sigma^2}{90} = 12.6$$

$$\sigma_{series} = \sqrt{Var(y_i)} = 3.55$$

A time series with this amount of noise and the value for μ estimated from the time series in figure 5.1 has been simulated and is shown in figure 5.5. Similar to figure 5.2 the periods have been averaged in figure 5.6.

Detection can of course be improved by recording more cycles of rest and applied stimuli, but due to cost considerations about 5 minutes is spent for each recording.

To get an idea of how knowledge of prior probabilities effects the error rate we make calculations assuming a prior probability of 0.9 for a voxel not to be in the activated state. First we have to calculate the optimum threshold from (2.6). The threshold is increased from $\frac{\mu}{2} = 1.23$ to $\hat{d}_{thres} = 1.73$. α is calculated from (2.9) giving $\alpha = 0.0104$, i.e. Type I Errors are reduced by 79%. $1 - \beta$ can be calculated from (2.10) giving $1 - \beta = 0.1643$, i.e. Type II Errors are increased by 229%. From (2.11) the average error rate is $\alpha p + (1 - \beta)(1 - p) = 0.0104 \cdot 0.9 + 0.1643 \cdot (1 - 0.9) = 0.0258$, i.e. average error rate is reduced by 48%.

The shape of the histogram of voxel data with parameter values resulting in this error rate is given in figure 5.7. To have the possibility of classifying the voxels by using our simple model, the histogram must separate the populations into two peaks. The probabilities of classification errors are determined by the amount of overlap between the two peaks.

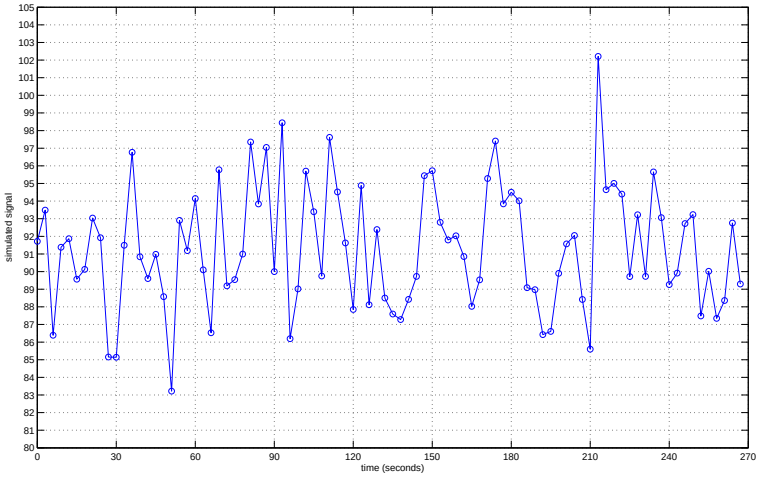

Figure 5.5: Time series for voxel simulated with parameters μ and σ_{series} giving an error rate of 0.05 when using our test statistic $\hat{d}$.

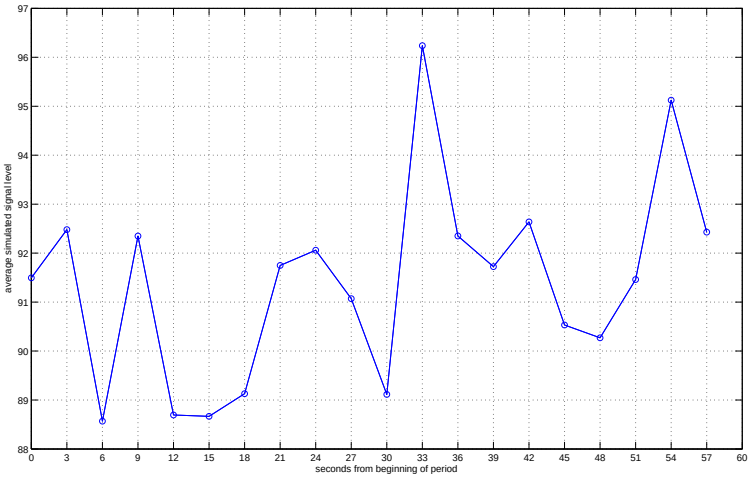

Figure 5.6: Periods of simulated time series in figure 5.5 averaged.

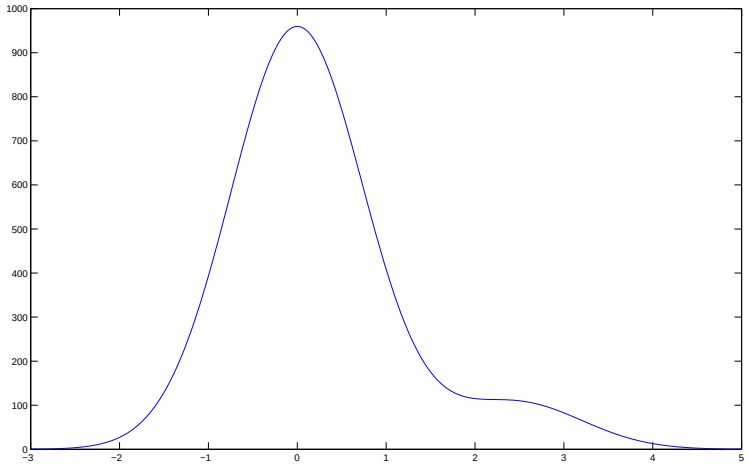

Figure 5.7: Shape of histogram for the test statistic $\hat{d}$ with $p = 0.9$ and parameters μ and σ giving a misclassification rate of 0.05 without prior probabilities considered. Misclassification rate is reduced to 0.026 if p is taken into account for classification.

Chapter 6

Data from fMRI recording

In this chapter we take a look an fMRI recording. No preprocessing except motion correction has been applied to the data, we have been told. This sort of correction is always applied to fMRI recordings because the brain moves in accord with the heartbeats.

6.1 Data format

The recording is a 4-dimensional 96x96x29x117 data set. The first three coordinates, which we name x, y, and z respectively, are space coordinates while the fourth coordinate, named t, is a time coordinate. Thus we have a discrete time series for each voxel containing 117 equally spaced samples.

The measured values have been digitized into one of 1256 digital values. This fact is easily established by sorting the values, storing them in a 1-dimensional array and doings searches in this array. Furthermore, for some unknown reason, the recorded values have been multiplied by a strange constant (1001.1473...) obviously (precise multiples of this number) in a digital circuit. For our analysis the measured values are more tractable by our computer software if not multiplied by this constant. So we have initially divided all the values in our data set by this constant to remove the scaling factor.

Together with the recording we have been supplied with a mask for sorting out the voxels picturing gray brain matter, i.e. the part of the brain of interest for fMRI. The mask selects 21,187 voxels of a total of 96x96x29 = 267,264 voxels comprising the entire 3-dimensional moving picture.

The time series seems to be periodic with 18 discrete values recorded every period of the recording time. An example of such a time series for a voxel is shown in figure 6.1. Taking a look at the time series for several voxels shows that the first 3 samples of the series have been recorded before the alternations between rest and applied stimuli have begun. These 3 samples could have been used for calculating the mean value during periods of rest. But to avoid patching written software, the time series have been truncated to the last 114 samples.

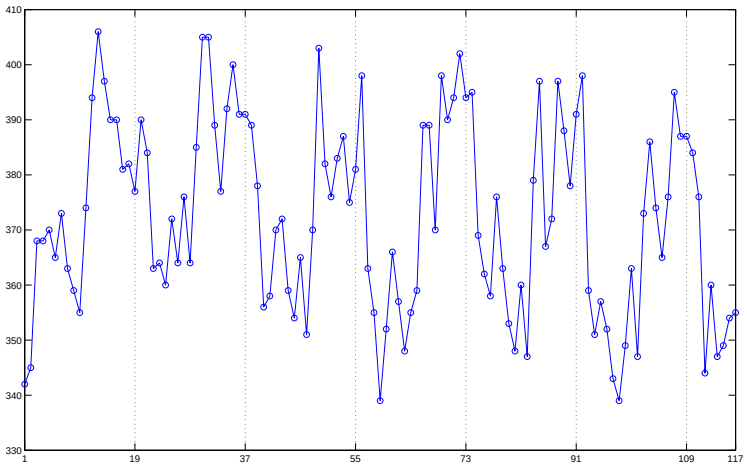

Figure 6.1: Example of time series in the data set. After calculating our test statistic and sorting in ascending order, this is the time series with index 21150, which means that only 37 time series in the recording show activation more clearly than this one.

6.2 Noise introduced by digitizing the measured values

The sampled values from the volume of the brain containing gray brain matter spans from 244 to 672, i.e. 429 integers. This digitization introduces noise, but this noise is quite small compared to the noise from other sources. If this noise is taken into account, another noise term must be added in (5.5). If all values of the fractional part of the rounded measured value are considered equally likely, this additional noise term has a uniform distribution. The probability model for the sampled values of the time series now become $y_i = x_i + \varepsilon_N + \varepsilon_U,\ i = 1, 2, \ldots 114,\ \varepsilon_N \sim N(0, \sigma^2_{series}),\ \varepsilon_U \sim U(-0.5, 0.5)$. As later pointed out, for this particular recording we obtain our test statistic by averaging 54 values, then averaging 48 values and calculating the difference between the averages. The variance of a uniformly distributed random variable on a unit interval is $\frac{1}{12}$, so the noise term in our test statistic $\hat{d}$ resulting from ε_U has variance $(\frac{1}{54} + \frac{1}{48}) \cdot \frac{1}{12} = 0.0033$. Due to the Central Limit Theorem the resulting noise term will have a distribution close to normal. Figure 6.2 shows the result of a simulation.

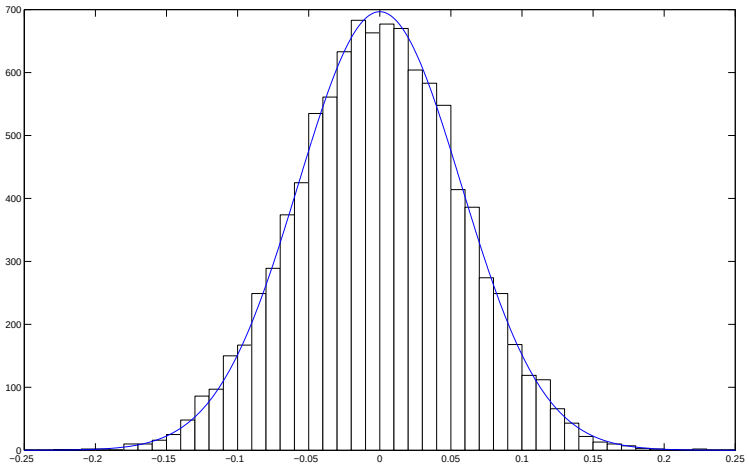

Figure 6.2: Histogram of 10,000 calculations of $\frac{1}{102}\sum_{i=1}^{102}\varepsilon_{U,i}$ compared to an $N(0, 0.033)$ distribution.

6.3 Calculating the test statistic

After an inspection of the time series for several evidently activated voxels we have chosen to calculate the test statistic by averaging samples 9-17, 27-35, ... , averaging samples 1-7, 19-25, ... and subtracting the latter average from the former. Looking at figure 6.4 we see that the level changes between samples 7 and 9 and after sample 17. The average levels of samples 8 and 18 are far less than half the difference between the average in the nonactivated state and the overall average of the series. Including these samples will give a less precise test statistic due to the added noise. So the chosen expression for the test statistic for this fMRI recording is

$$\hat{d} = \bar{y}_{activity} - \bar{y}_{rest} \tag{6.1}$$

$$= \frac{1}{54}\sum_{j=0}^{5}\sum_{i=9}^{17} y_{i+18j} - \frac{1}{48}(\sum_{j=0}^{6}\sum_{i=1}^{6} y_{i+18j} + \sum_{j=0}^{5} y_{7+18j}) \tag{6.2}$$

The resulting histogram for the test statistic for the 21187 voxels showing gray brain matter with the capability of being in an activated state is shown in figure 6.5. One reason for the bad fit between our model and the real data is that the distribution has a long tail to the right. This is at least in part due to the properties of the gray brain matter. The strength of the measured signal from activated parts of the brain is not constant as assumed in our simple model.

Figure 6.5 also shows the result of estimating the MLE's for the density

function

$$f(\hat{d};\sigma_0^2,\sigma_1^2,\mu_0,\mu_1,p)=\frac{p}{\sqrt{2\pi\sigma_0^2}}e^{-\frac{(\hat{d}-\mu_0)^2}{2\sigma_0^2}}+\frac{1-p}{\sqrt{2\pi\sigma_1^2}}e^{-\frac{(\hat{d}-\mu_1)^2}{2\sigma_1^2}} \tag{6.3}$$

If $\mu_0 = 0$ and $\sigma_0^2 = \sigma_1^2$ this is our simple model from (2.14). Estimation of the parameters setting $\sigma_0 = \sigma_1$ or $\mu_0 = 0$ has been done in addition to estimation for our proposed simple probability model (2.14) and the normal-Bernoulli model with all applicable parameters included as given in (6.3).

As expected the density function (6.3) with most parameters (5) to be estimated gives the best approximation and the density function with fewest parameters (3) to be estimated gives the poorest approximation to the histogram and the other two alternatives give something in between. But the heavy deviation of the histogram from the estimated functions indicates that our assumptions for proposing the probability moedel are not valid. As formerly seen in figure 6.2 even probability distributions far from normal in shape as the uniform distribution produce a distribution close to normal when independent identically distributed samples are summed. The way our test statistic is derived should guarantee that within the two populations the test statistic has a close to normal distribution.

6.4 Testing the validity of our model

6.4.1 Goodness-of-Fit Test

To assess the severity of the deviation of the histogram from the density functions we can do a Goodness-of-Fit Test as described in "Larsen and Marx: An Introduction to Mathematical Statistics and Its Applications", third edition, page 540. Our test statistic to be calculated for performing this test is

$$c_1=\sum_{i=1}^{k}\frac{(x_i-n\hat{p}_i)^2}{n\hat{p}_i} \tag{6.4}$$

It is best making the division into intervals for performing the test coincide with our columns in the histogram in figure 6.5. Then k becomes the number of histogram columns included in this hypothesis test (the columns should be chosen such that $n\hat{p}_i \geq 5$ for all k columns), x_i is the number of values of the calculated test statistic included in column i and n is the number of samples drawn from the distribution, in this case n = 21,187, the number of voxels showing gray brain matter. $\hat{p}_i$ is the estimated probability for the test statistic making it one of the values of column i (depends on our choice of probability model and the estimated parameters). With our choice of model $\hat{p}_i$ is calculated as

$$\hat{p}_i=\hat{p}(\Phi(\frac{d_{i+1}-\hat{\mu}_0}{\hat{\sigma}_0})-\Phi(\frac{d_i-\hat{\mu}_0}{\hat{\sigma}_0}))+(1-\hat{p})(\Phi(\frac{d_{i+1}-\hat{\mu}_1}{\hat{\sigma}_1})-\Phi(\frac{d_i-\hat{\mu}_1}{\hat{\sigma}_1})) \tag{6.5}$$

$\Phi(\cdot)$ is the standard normal cumulative distribution function. d_i is the left border of column i of the histogram of the calculated test statistics.

$$c_1 \sim \chi^2_{k-1-r}$$

r is the number of estimated parameters. $r = 3$ for our simple model, $r = 5$ for the normal-Bernoulli model with all applicable parameters included and $r = 4$ if we restrict the latter to $\mu_0 = 0$ or $\sigma_0 = \sigma_1$. The hypothesis that the data is from the model assumed while doing the calculations in (6.5) should be rejected with significance level α if $c_1 \geq \chi^2_{1-\alpha,k-1-r}$.

6.4.2 Result while testing our probability models

The test rejects the hypothesis that the histogram is drawn from a normal-Bernoulli distribution.

While testing our simple model using all the columns with expectation greater than 5 (i.e. the columns with $\hat{d}$ from -11.4 to 21), we obtain $c_1 \approx 3,284$ for a χ^2_{132} test statistic. The P-value for this value of the test statisticis truncated to 0 by the computer software, so it is almost quite impossible that our model is applicable.

If the test is performed for the model allowing the main peak to be displaced from 0, the columns from -9.72 to 23.16 can be included. We obtain $c_1 \approx 1,419$ for a χ^2_{133} test statistic and again the P-value is truncated to 0.

If the test is performed for the model allowing for different variances for the 2 populations but not allowing for the main peak to be displaced from 0, the columns from -10.68 to 20.28 can be included. We obtain $c_1 \approx 1,391$ for a χ^2_{125} test statistic and again the P-value is truncated to 0.

If the test is performed for the normal-Bernoulli model with all applicable parameters, the columns from -10.44 to 22.68 can be included. We obtain $c_1 \approx 386$ for a χ^2_{133} test statistic and still the P-value is truncated to 0.

6.5 Looking at the noise in the fMRI time series

So far in this chapter we have estimated the parameters the way it is usually done for pictures consisting of two main parts of interest. For the fMRI data we have a time series available for each voxel. As shown in section 5.3 the noise in each voxel can be estimated from its time series.

6.5.1 Distribution of the sample variance

For this particular fMRI recording, after cutting away the first three samples of the time series, the expression for S_p^2 is

$$S_p^2 = \frac{1}{6\cdot 6 + 5\cdot 12}(6(S_1^2 + \ldots + S_6^2) + 5(S_7^2 + \ldots + S_{18}^2)) = \frac{1}{96}(6\sum_{i=1}^{6} S_i^2 + 5\sum_{i=7}^{18} S_i^2)$$

$$= \frac{1}{96}(6\sum_{i=1}^{6}\frac{1}{6}\sum_{j=0}^{6}(y_{i+18j} - \bar{y}_i)^2 + 5\sum_{i=7}^{18}\frac{1}{5}\sum_{j=0}^{5}(y_{i+18j} - \bar{y}_i)^2)$$

$$= \frac{1}{96}(\sum_{i=1}^{6}\sum_{j=0}^{6}(y_{i+18j} - \bar{y}_i)^2 + \sum_{i=7}^{18}\sum_{j=0}^{5}(y_{i+18j} - \bar{y}_i)^2) \tag{6.6}$$

$$\bar{y}_i = \frac{1}{7}\sum_{j=0}^{6} y_{i+18j} \qquad i = 1, \ldots, 6 \tag{6.7}$$

$$\bar{y}_i = \frac{1}{6}\sum_{j=0}^{5} y_{i+18j} \qquad i = 7, \ldots, 18 \tag{6.8}$$

$$\frac{96S_p^2}{\sigma_{series}^2} \sim \chi_{96}^2$$

$$S_p^2 \sim \frac{96}{\sigma_{series}^2}\ \chi_{96}^2(\frac{96s_p^2}{\sigma_{series}^2}) \tag{6.9}$$

S_p^2 should have a χ_{96}^2 distribution close to a normal distribution in shape (again due to the CLT) with variance $\frac{2\sigma^4}{96}$. Calculating the S_p^2's and plotting a histogram shows that this is far from true. The histogram is far from a χ_{96}^2 distribution in shape. The parameter σ_{series}^2 in distribution (6.9) has been estimated by using maximum likelihood and the resulting probability distribution plotted. The largest value of S_p^2 is 7,863, i.e. the tail to the right is about ten times as long as is shown in figure 6.6. The next largest value is $S_p^2 \approx$ 5,723, there are 4 values greater than 5,000, 16 values greater than 2,000 and 71 values greater than 800 (i.e. not included in the histogram). It was necessary to exclude the largest 19 values (1,663 the largest value of S_p included in the MLE calculaton) while estimating σ_{series}^2 to avoid getting a zero argument for the log function while calculating $log(\frac{96}{\sigma^2}\ \chi_{96}^2(\frac{96S_{p,i}^2}{\sigma^2})$ in the numerical maximization of the likelihood function. The value obtained for the maximum likehood estimator was $\hat{\sigma}_{series}^2 = 87.6$.

6.5.2 Calculating the variance of the test statistic from the variance of the time series

For calculating the noise of our test statistic from σ_{series}^2 we obtain an equation similar to equation (5.4). Taking the variance of expression (6.2) gives

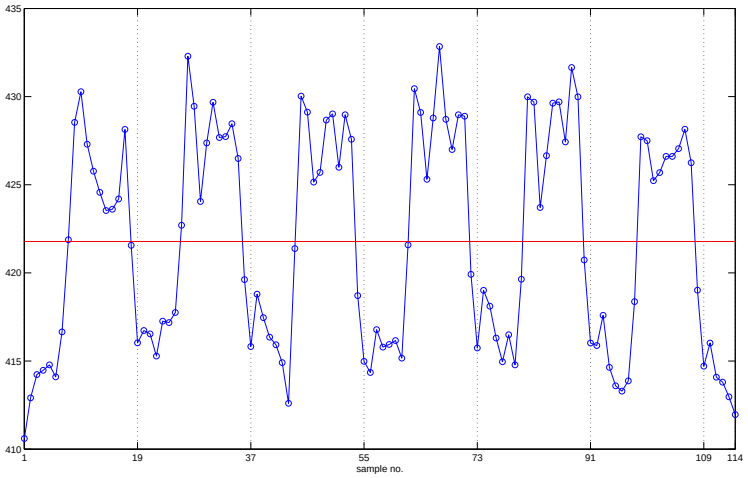

Figure 6.3: Average of the time series for the 100 voxels having test statistic from 12.01 to 12.75.

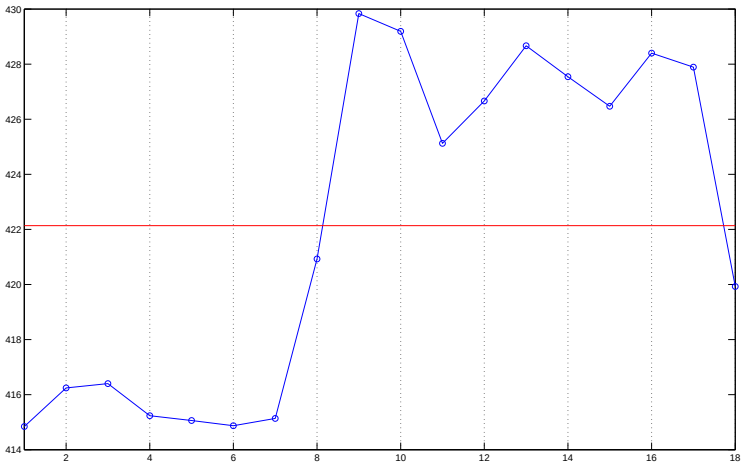

Figure 6.4: Average of the periods in figure 6.3.

$$\sigma = Var(\hat{d}) = Var(\frac{1}{54}\sum_{j=0}^{5}\sum_{i=9}^{17} y_{i+18j} - \frac{1}{48}(\sum_{j=0}^{6}\sum_{i=1}^{6} y_{i+18j} + \sum_{j=0}^{5} y_{7+18j}))$$
$$= \frac{1}{54^2}\sum_{j=0}^{5}\sum_{i=9}^{17} Var(y_i) + \frac{1}{48^2}(\sum_{j=0}^{6}\sum_{i=1}^{6} Var(y_i) + \sum_{j=0}^{5} Var(y_i))$$
$$= \frac{1}{54^2}\,54Var(y_i) + \frac{1}{48^2}\,48Var(y_i) = \frac{51}{1296}\,\sigma^2_{series} \tag{6.10}$$

Taking our maximum likelihood estimation of σ^2_{series} as the true value for the variance of the time series in our model gives

$$\sigma = \sqrt{\frac{51}{1296}\,87.6} = 1.857 \tag{6.11}$$

for the variance of our test statistic, a smaller value than the values of all of the estimated σ's from the histogram of the test statistic. This is reasonable because the spread of the two populations is partly due to other causes than random noise in the samples. But the problem this estimation also is that the assumption of equally distributed noise does not hold.

A scatterplot of the values of the test statistic and the noise is shown in figure 6.8 and figure 6.9

6.5.3 Looking for correlation of noise in adjacent voxels

The correlation coefficient is defined as

$$\rho(X,Y) = \frac{Cov(X,Y)}{\sqrt{Var(X)}\sqrt{Var(Y)}} = \frac{E(XY) - E(X)E(Y)}{\sqrt{Var(X)}\sqrt{Var(Y)}} \tag{6.12}$$

Usually it is estimated by calculating the sample correlation coefficient, but to eliminate the periodic variations of the time series we estimate it by using a technique similar to the estimation (6.6) of the noise of the time series. Here the variances in (6.12) is estimated as the mean of the variances of $y_i,\ i = 1, 2, \ldots$ (similarly for Var(X)):

$$\hat{Var}(Y) = \frac{1}{114}(\sum_{i=1}^{6}\sum_{j=0}^{6}(y_{i+18j} - \bar{y}_i)^2 + \sum_{i=7}^{18}\sum_{j=0}^{5}(y_{i+18j} - \bar{y}_i)^2) \tag{6.13}$$

Here the covariance of the noise is estimated as

$$\hat{Cov}(X,Y) = \frac{1}{114}(\sum_{i=1}^{6}\sum_{j=0}^{6} x_{i+18j}\ y_{i+18j} - 7\sum_{i=1}^{6}\bar{x}_i\bar{y}_i + \sum_{i=7}^{18}\sum_{j=0}^{5} x_{i+18j}\ y_{i+18j} - 6\sum_{i=7}^{18}\bar{x}_i\bar{y}_i) \tag{6.14}$$

The correlation coefficients have been estimated between neighbours in the x, y

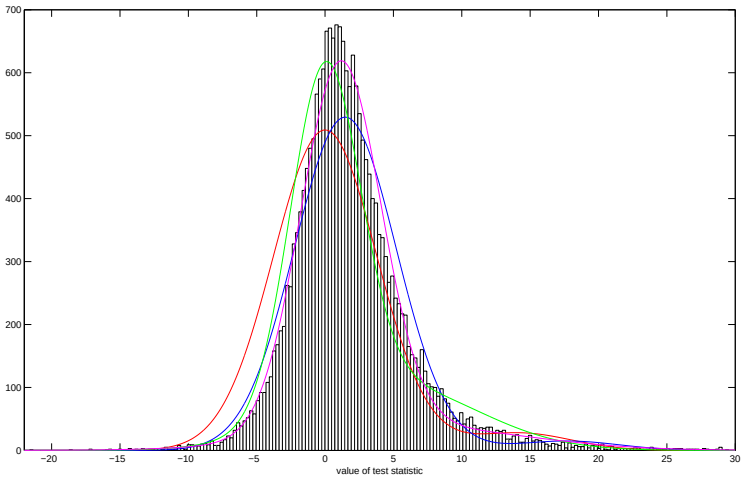

Figure 6.5: Histogram of our test statistic calculated from the given fMRI recording. The red curve is the resulting density function when applying our simple model. The estimated MLE parameters for this model are $p = 0.948$, $\mu = 14.2$ and $\sigma = 3.78$ (red curve). A better fit is obtained if the simple model is modified, allowing for different variances for the two populations of voxels (nonactivated and activated). Estimating the MLE parameters for this modified model gives $p = 0.679$, $\mu = 4.85$, $\sigma_0 = 2.55$ and $\sigma_1 = 6.30$ (green curve). If allowing for the main peak to be located away from 0, but assuming the same variance for the two populations, we obtain $p = 0.972$, $\mu_0 = 1.48$, $\mu_1 = 17.7$ and $\sigma = 3.73$ for the MLE's (blue curve). Allowing for both the main peak being location away from 0 and for different variances for the two populations $p = 0.847$, $\mu_0 = 1.15$ and $\mu_1 = 6.24$, $\sigma_0 = 2.92$ and $\sigma_1 = 8.25$ are obtained for the MLE's (magenta curve).

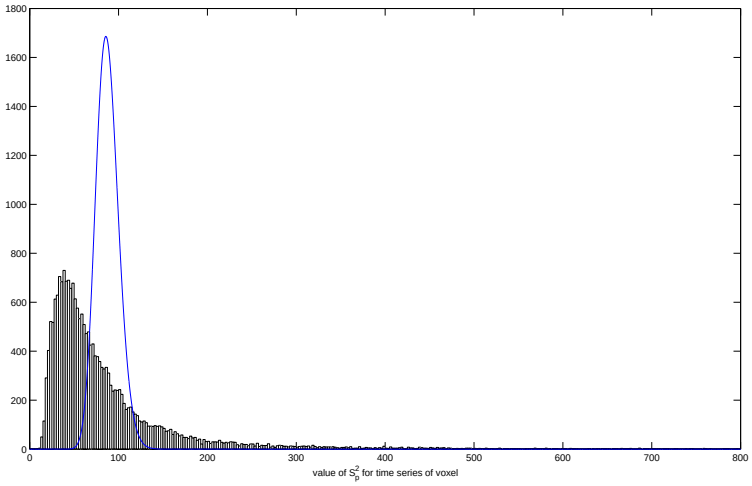

Figure 6.6: Histogram of pooled sample variance from the time series for the voxels showing gray brain matter.

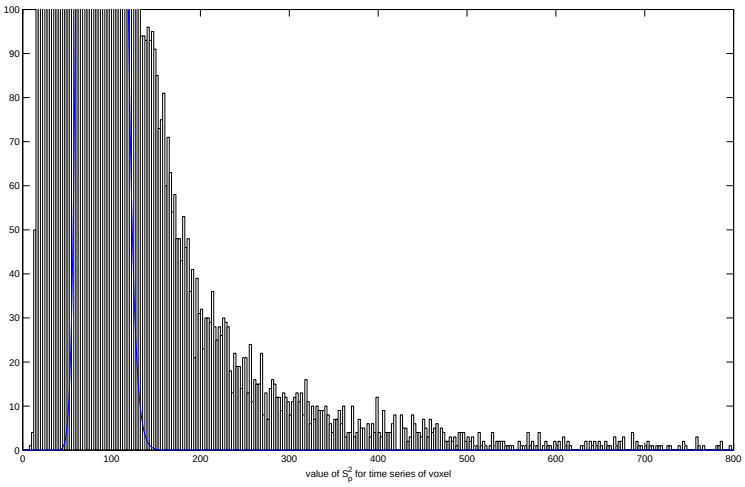

Figure 6.7: The same as figure 6.6 but vertical axis enlarged to better show the tail.

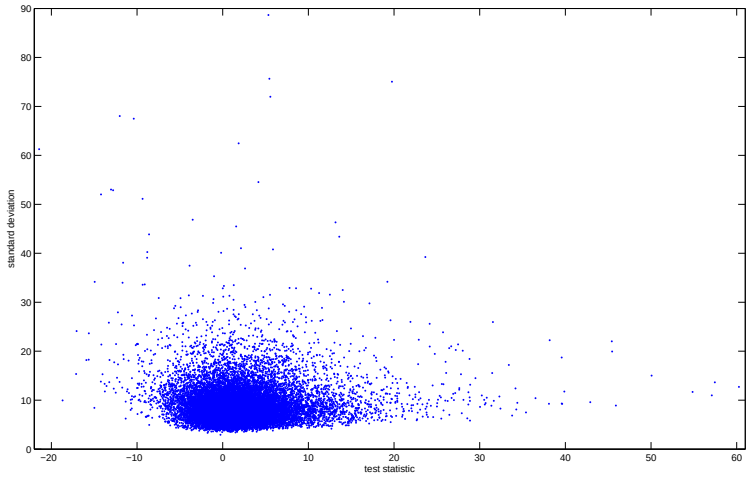

Figure 6.8: Scatterplot of the test statistics and their noise calculated from the time series.

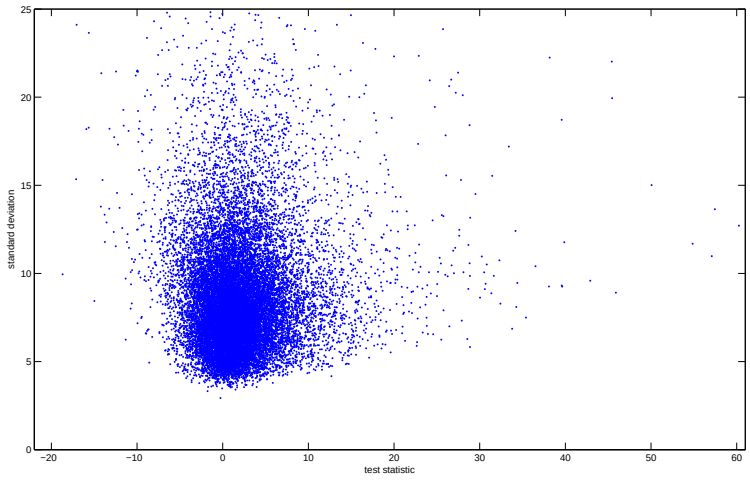

Figure 6.9: Scatterplot enlarge along the axis for the noise value.

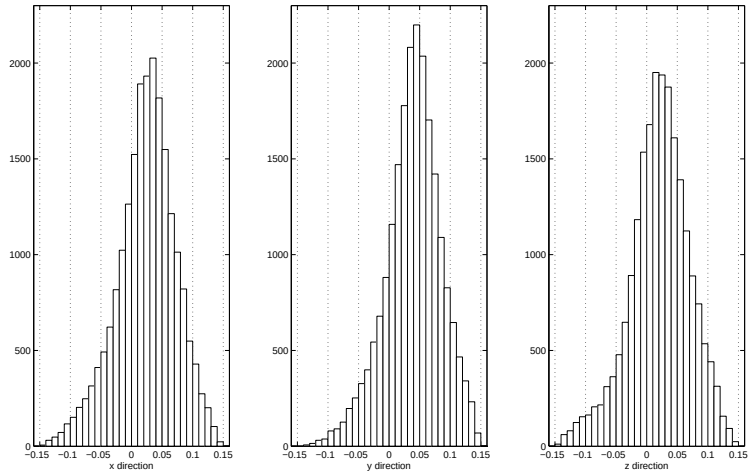

Figure 6.10: Histograms from estimation of correlation between noise in adjacent voxels in the x, y and z directions respectively in gray brain matter. The same correlation pattern applies in the other areas of the brain.

and z directions. There is some correlation between the noise of the neighbours. For some unknown reason the correlation is highest between neighbours in the y direction. See figure 6.10.

6.5.4 Examples of noise in the time series

6.5.5 Density distribution of the noise

The great variations in magnitude of the noise is surprising. But far from surprisingly due to the CLT the noise for a given voxel seems to be close to normal in shape. Histograms of the samples in the time series showing a test statistic $\hat{d}$ between -0.5 and 0.5 has been standardized by subtracting the mean and dividing the difference by the sample standard deviation. Then they have been added to obtain an average. The time series should contain very little or no periodic variations because of the value of the test statistic. If the noise in a given voxel is Gaussian the sum of the histograms should be close to a Student t distribution with 114 degrees of freedom in shape (i.e. also very close to a normal distribution). The result is shown in figure 6.13.

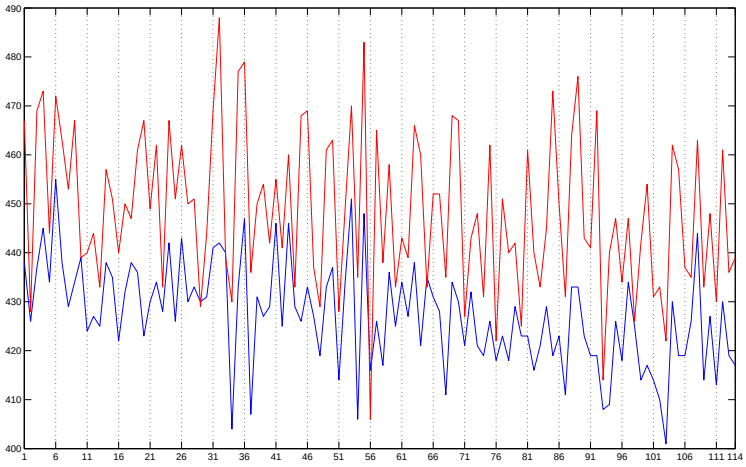

Figure 6.11: The noise from these adjacent voxels seems a bit correlated. Time series (blue) for voxel (53, 27, 3) with index 20000 when sorted with respect to estimated correlation coefficients and time series (red) for one of its closest neighbours (53, 28, 3) in the y direction. Mean value for the latter has been shifted by -105 for ease of comparison. Sometimes there is a large change in the mean value of the time series when moving between close neighbours. A histogram of the changes in mean values resembles a normal distribution with a standard deviation around 35.

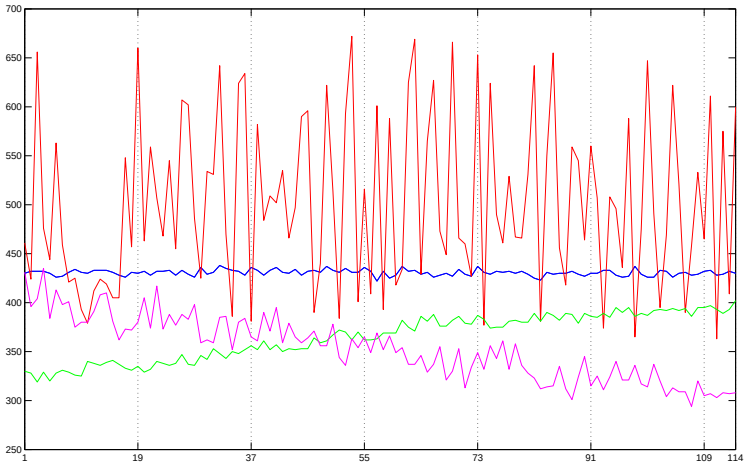

Figure 6.12: Time series for some voxels located in the gray brain matter. Time series for the most noisy voxel (50, 37, 28), the least noisy (42, 19, 18), voxel showing increasing average (28, 22, 10) and voxel showing decreasing average (24, 29, 13). The values of the test statistic $\hat{d}$ for these voxels are respectively 0.26, -0.27, 3.09 and -11.70. The drift of the average level falsely influences our chosen test statistic.

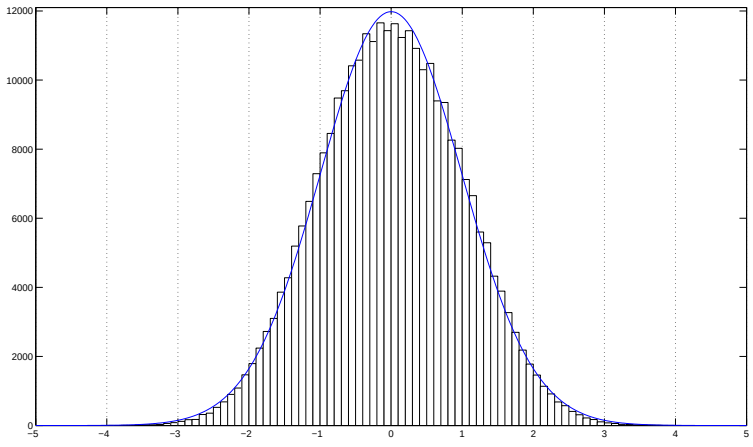

Figure 6.13: Noise appears to be close to normal in shape but differs in magnitude. Figure shows the average shape of the histograms of the samples in the 2640 time series exibiting $\hat{d}$ between -0.5 and 0.5. The shape of the Student t distribution with 114 degrees of freedom has been plotted for comparison.

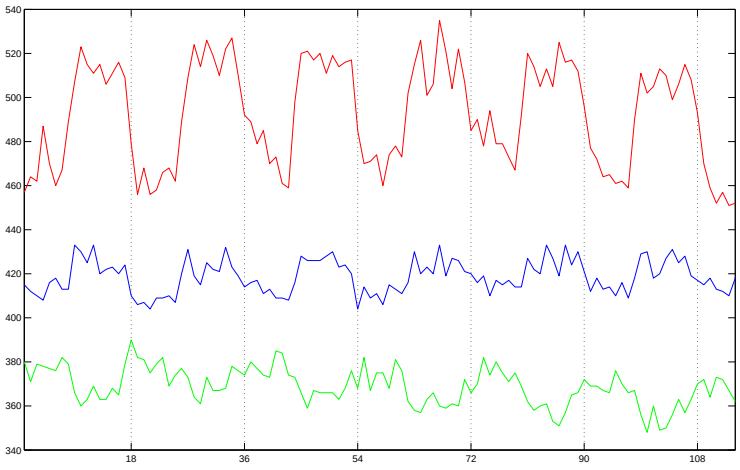

Figure 6.14: Voxels having different values for the test statistic. Time series for voxel (32, 43, 24) giving $\hat{d} = 45.9$ (red), voxel (25, 45, 16) giving $\hat{d} = 12.7$ (blue) and voxel (48, 18, 12) giving $\hat{d} = -11.3$ (green). The latter is an example of a voxel being deactivated by the applied stimuli. (Mean of uppermost time series shifted +100 and mean of lowermost time series shifted by -20 before plotting.)

Chapter 7

Obtaining better separation of the voxels

7.1 Separation methods from chapter 5

7.1.1 Error expectations for our fMRI recording without taking prior probabilities into account

Using a standard hypothesis test for determining the state of a voxel results in error rates given by (2.7) and (2.8). Taking into account that σ_0 and σ_1 have different values we obtain the threshold of $\hat{d}$ by solving $f(\hat{d}|H_0) = f(\hat{d}|H_1)$. We obtain 5.376 for the threshold. If Φ is the standard normal cumulative distribution, the probability for a Type I error given H_0 is

$$P(Type\ I\ error|H_0) = 1 - \Phi(\frac{5.376 - \mu_0}{\sigma_0}) = 1 - \Phi(\frac{5.376 - 1.152}{2.924}) = 0.0743$$

The probability for a Type II error given H_1 is

$$P(Type\ II\ error|H_1) = \Phi(\frac{5.376 - \mu_1}{\sigma_1}) = \Phi(\frac{5.376 - 6.236}{8.255}) = 0.459$$

The expected classification error rate is

$$\begin{aligned} P(error) &= p \cdot P(Type\ I\ error|H_0) + (1-p) \cdot P(Type\ II\ error|H1) \\ &= 0.8473 \cdot 0.0743 + (1 - 0.8473) \cdot 0.459 = 0.133 \end{aligned}$$

7.1.2 Error expectations taking the average prior probability into account

If the average prior probability of a voxel being not activated, our parameter p, is taken into account, the error rate can be improved. Now we obtain the threshold by solving $pf(\hat{d}|H_0) = (1-p)f(\hat{d}|H_1)$. We obtain 8.041 for the threshold.

Otherwise the errors are calculated by using exactly the same expressions. Now the probability for a Type I error given H_0 is

$$P(Type\ I\ error|H_0) = 1 - \Phi(\frac{8.041 - \mu_0}{\sigma_0}) = 1 - \Phi(\frac{8.041 - 1.152}{2.924}) = 0.00924$$

In this case the probability for a Type II error given H_1 is

$$P(Type\ II\ error|H_1) = \Phi(\frac{8.041 - \mu_1}{\sigma_1}) = \Phi(\frac{8.041 - 6.236}{8.255}) = 0.587$$

And the overall expected error rate by assuming a constant prior probability is

$$\begin{aligned} P(error) &= p \cdot P(Type\ I\ error|H_0) + (1-p) \cdot P(Type\ II\ error|H_1) \\ &= 0.8473 \cdot 0.00924 + (1 - 0.8473) \cdot 0.587 = 0.0974 \end{aligned}$$

Taking p into account can dramatically improve the error rate for the voxels being in the most likely state but the error rate for the voxels in the least likely state increases. The net result is an improvement in the overall error rate.

7.2 Improved Bayesian restoration

So far we have used Bayesian restoration in a very simple manner: Assuming that the prior probability for a voxel belonging to either of the two populations is constant. This can be approved upon: The prior probability of a voxel being in a given state can be chosen as a function of the state of the nearby voxels.

7.3 Use of the Geman McClure potential

The Geman McClure potential is given by expression (3.9). The ICM algorithm for use with fMRI recordings has been implemented using this potential with parameters $\beta = 0.7$ and $\delta = 1.2\sigma_0 = 1.2 \cdot 2.92$. The initial histogram of the test statistic and the histogram after the first iteration is shown in figure 7.1. Figures 7.2, 7.3 and 7.4 shows the histogram after further iterations. As can be seen the separation of the peaks needs only a few iterations.

7.4 Using ICM two sort directly into two populations

It is reasonable to assume that $P(x_i = \mu_0|x_{\partial i})$ will approximate a function given by a falling straight line starting at $P(x_i|\text{no neighbours activated}) = p_{max}$ and ending at $P(x_i|\text{all neighbours activated}) = p_{min}$. Maybe an S-shaped line with bent in either direction will be a better approximation. So we try some kind of S-shaped curve with a parameter for the bend, $P(x_i|\text{no neighbours activated})$ and $P(x_i|\text{all neighbours activated})$ as its three parameters. In the proposed

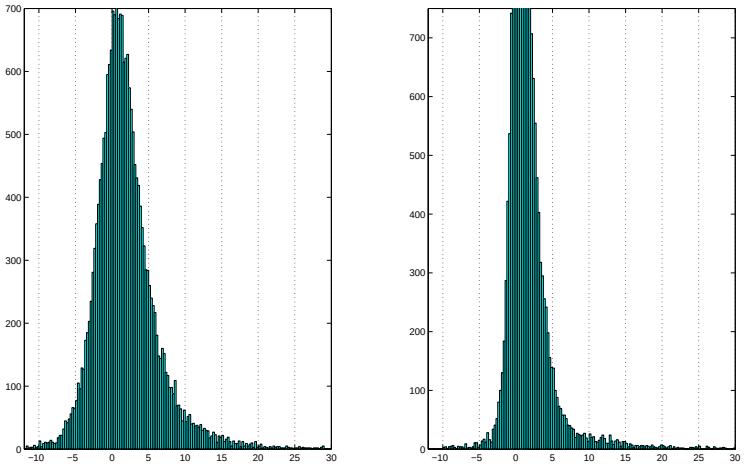

Figure 7.1: Initial histogram before running ICM and histogram after the first iteration.

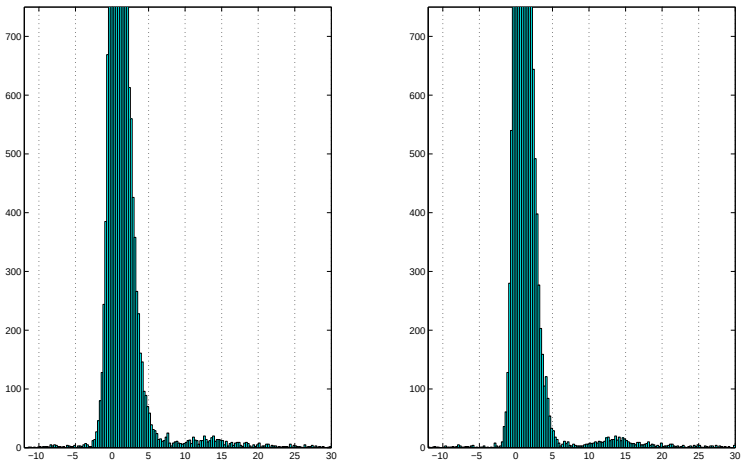

Figure 7.2: Histograms after the second and third iteration.

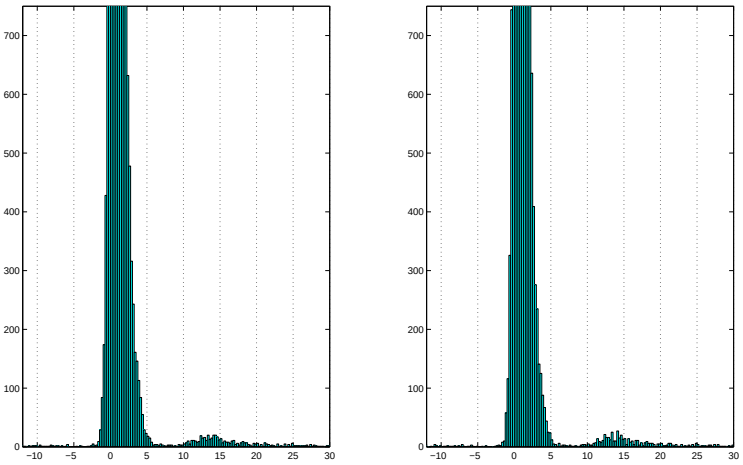

Figure 7.3: Histograms after the fourth and fifth iteration.

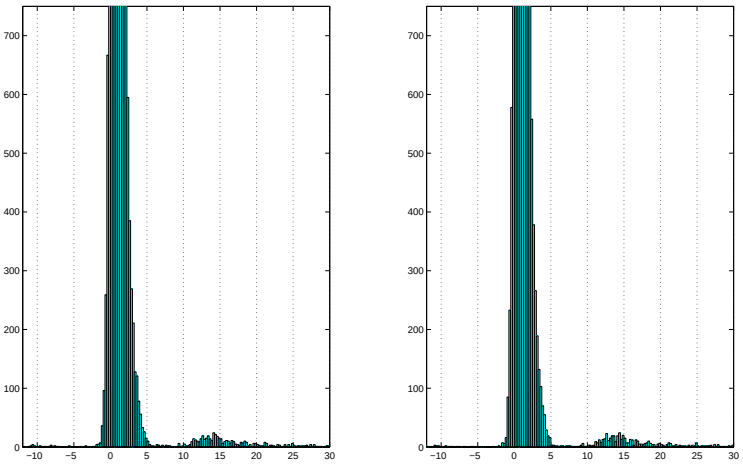

Figure 7.4: Histograms after the sixth and seventh iteration.

approximations for the true $P(x_i = \mu_0|x_{\partial i})$ we name these parameters b, p_{max} and p_{min} respectively.

A weighted sum of voxels in the nonactivated state is chosen as the argument of this function. We define the neighbourhood to be the set of the 26 voxels closest to voxel i. There is a first order neighbourhood of 6 voxels, one at each side of voxel i, a second order neighbourhood of 12 voxels, one at each line where two sides of the voxel meet, and a third order neighbourhood of 8 voxels, one at each corner where three sides meet. The neighbourhood voxels are given a weight inversely proportional to the distance of their centers from the center of voxel i.

For our fMRI recordings things are a bit more complicated because the resolution in the z direction is 4.0 mm and the resolution in the x and y direction is 2.4 mm, i.e. the resolutionin the z direction is $\frac{5}{3}$ times the resolution in the other two directions. So we use two different weights for voxels in the first order neighbourhood and also two different weights for voxels in the second order neighbourhood dependent upon whether the displacement of the neighbour voxel from voxel i includes displacement in the z direction. We use the weights 1 (4 voxels) or $\frac{3}{5} = 0.6$ (2 voxels) for voxels in the first order neighbourhood, $\frac{1}{\sqrt{1^2+1^2}} = \frac{1}{\sqrt{2}} \approx 0.71$ (4 voxels) or $\frac{1}{\sqrt{1^2+(\frac{5}{3})^2}} = \frac{1}{\sqrt{\frac{34}{9}}} \approx 0.51$ (8 voxels) for voxels in the second order neighbourhood and $\frac{1}{\sqrt{1^2+1^2+(\frac{5}{3})^2}} = \frac{1}{\sqrt{\frac{43}{9}}} \approx 0.46$ for all the voxels in the third order neighbourhood.

By counting the number of voxels given each of the weights above, multiplying by the weights and summing, a variable used as the argument for a function giving the probability $P(x_i = \mu_0|x_{\partial i})$ is obtained. In our case it is a number ranging from 0 to $4 \cdot 1 + 2 \cdot \frac{3}{5} + 4 \cdot \frac{1}{\sqrt{2}} + 8 \cdot \frac{3}{\sqrt{34}} + 8 \cdot \frac{3}{\sqrt{43}} \approx 15.8$. We refer to this weighted sum of activated neighbour voxels as w and its maximum value as w_{max}.

A straight line starting at p_{max} for $w = 0$ and ending at p_{min} for $w = w_{max}$ is given by

$$p(w) = \frac{(p_{min} - p_{max})w}{w_{max}} + p_{max} \qquad w \in [0, w_{max}]$$

A simple proposal for an S-shaped curve, symmetric about its midpoint, having the same endpoints as the straight line is

$$p(w) = \frac{1}{2}((p_{max} - p_{min})(1 + \frac{1}{bw})(\frac{1}{1+bw} - \frac{1}{1+b(w_{max}-w)}) + p_{min} + p_{max})$$

$$w \in [0, w_{max}]$$

A proposal for a curve having a bend in the opposite direction, still having the same endpoints is

$$p(w) = \begin{cases} \frac{1}{2}((p_{max} - p_{min})(1 + \frac{2}{bw_{max}})\frac{-b(w-\frac{w_{max}}{2})}{1-b(w-\frac{w_{max}}{2})} + p_{min} + p_{max}) & \text{if } 0 \le w \le \frac{w_{max}}{2} \\ \frac{1}{2}((p_{max} - p_{min})(1 + \frac{2}{bw_{max}})\frac{-b(w-\frac{w_{max}}{2})}{1+b(w-\frac{w_{max}}{2})} + p_{min} + p_{max}) & \text{if } \frac{w_{max}}{2} < w \le w_{max} \end{cases}$$

We note that the deactivated (or negative activated) voxels are classified into the population of activated voxels due to the larger variance of the test statistic for activated voxels. See figure 3.4. However it is easy to separate the deactivated from the activated voxels by just testing the sign of the test statistic for the activated population obtaining a classification into three populations: Nonactivated, activated and deactivated voxels.

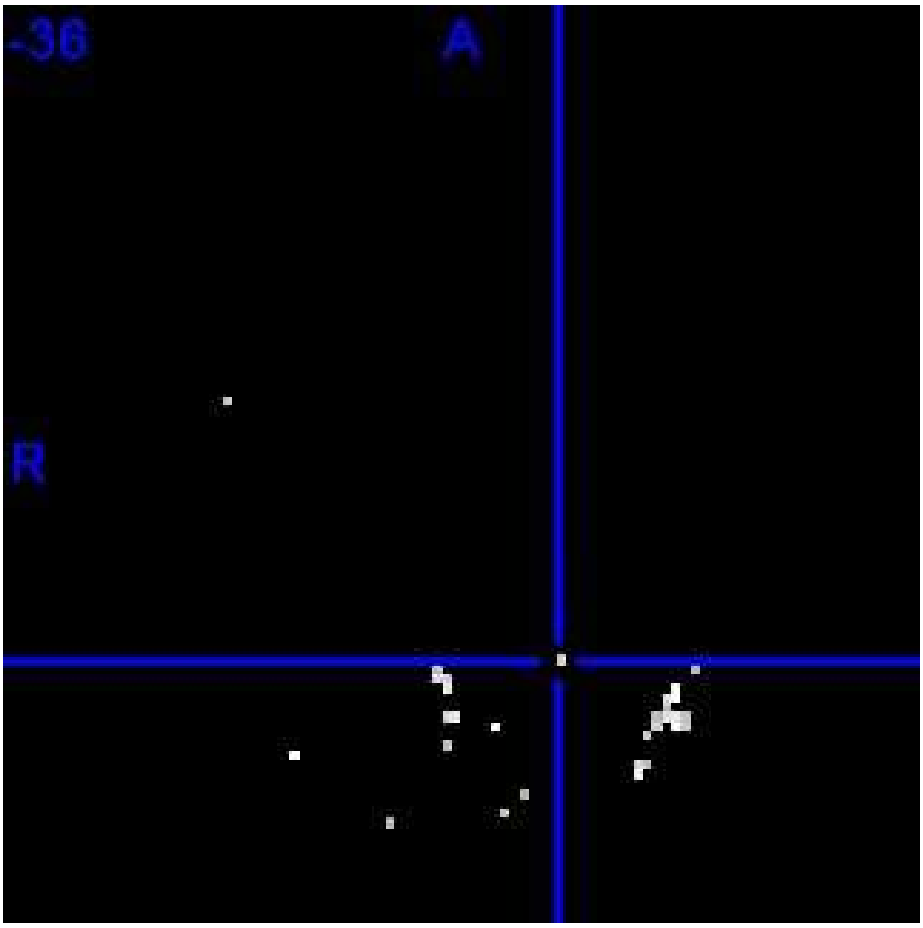

Figure 7.5: fMRI picture from the University Hospital

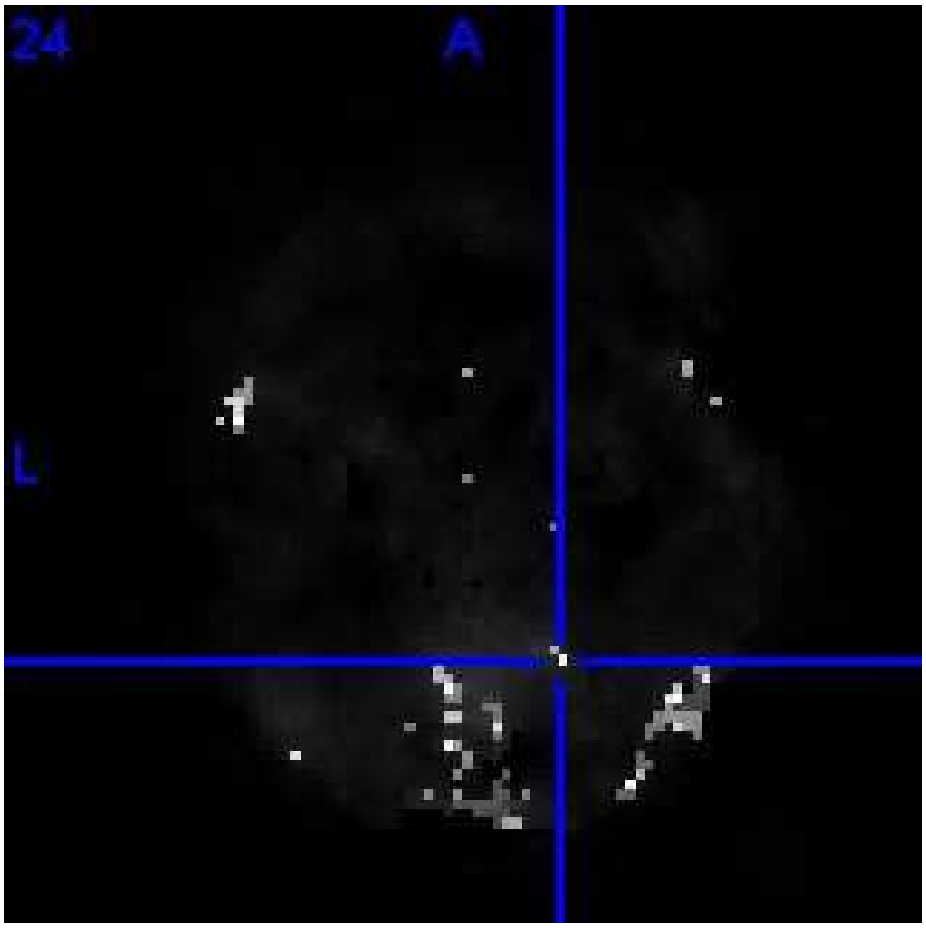

Figure 7.6: The same fMRI picture obtained by ICM

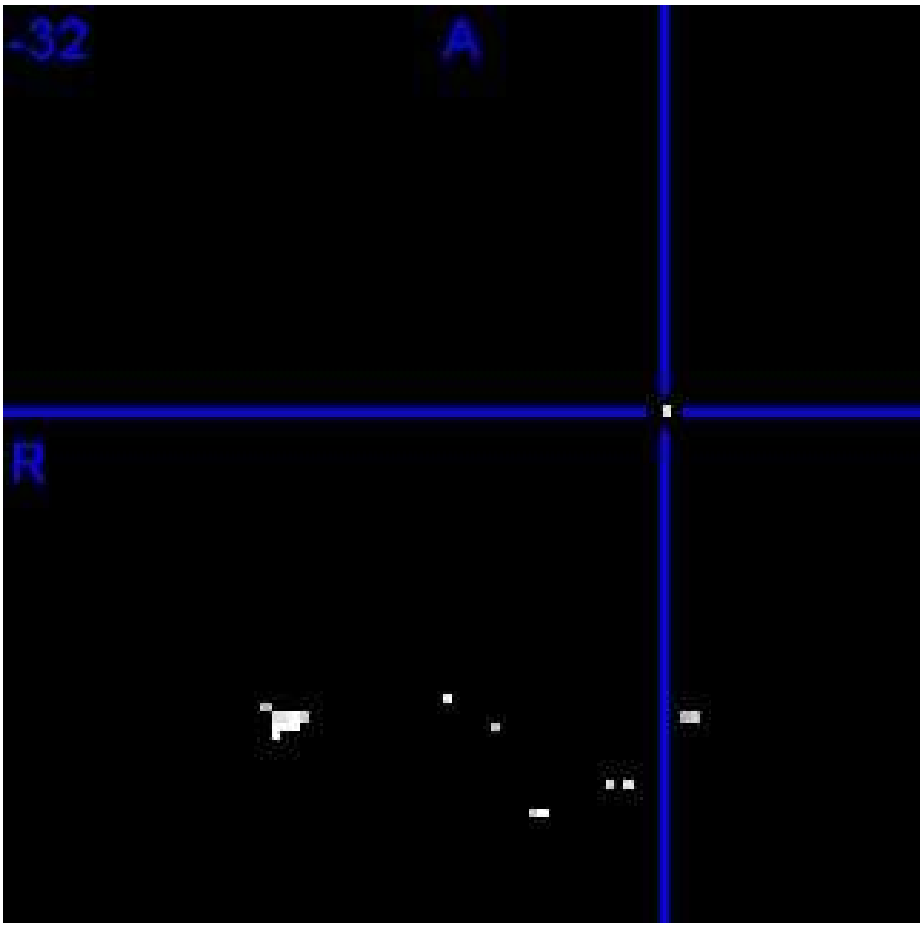

Figure 7.7: fMRI picture from the University Hospital

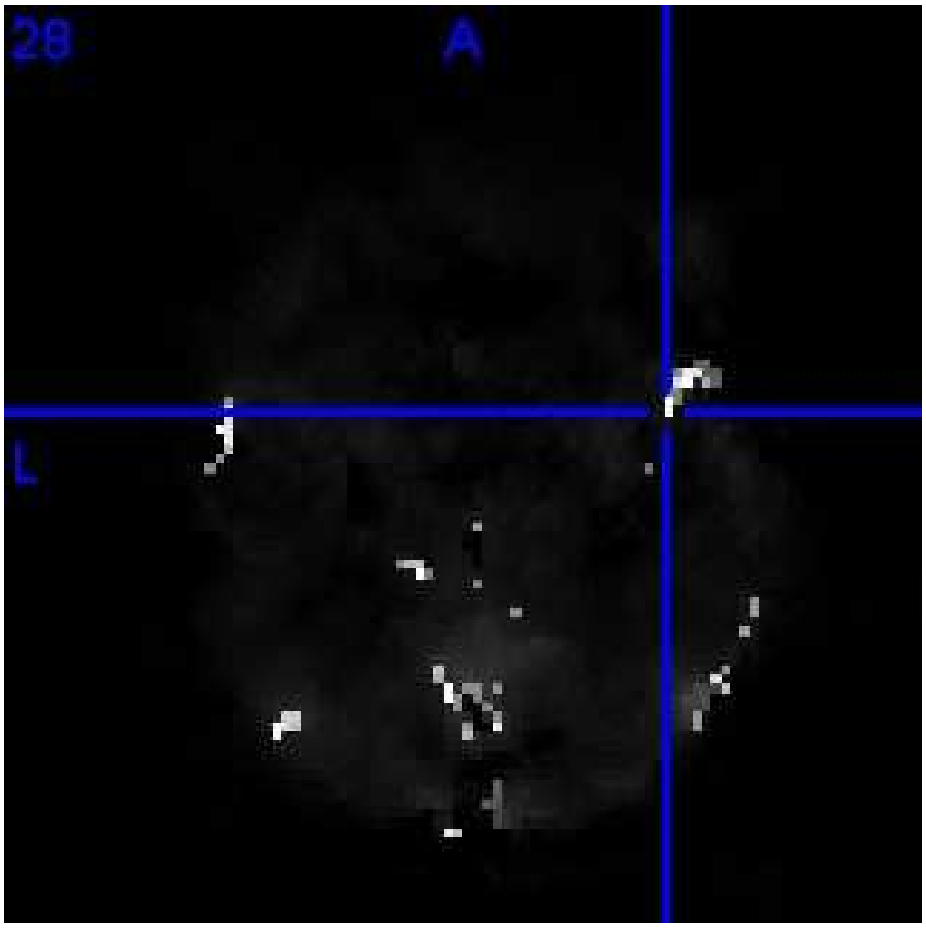

Figure 7.8: The same picture obtained by ICM

Chapter 8

Application of the ICM algorithm to find the border of photographed moles

The ICM algorithm has alsobeen tried as a method to draw the border of moles. This is part of a project for improving the process of detecting melanoma. Histogram of the test statistic is shown in figure 8.1 together with the distribution estimated by the use of maximum likelihood.

A picture with estimate of the border is shown in figure 8.2. The black curve is the border estimated from the use of ICM. The other lines are determined by methods developed by another student. A straight line for the prior probability starting at $p = 0.99975$ for no mole pixels in the neighbourhood and ending at $p = 0.00025$ if all the neighbourhood voxels show mole has been used. The estimated position of the border depends on the choice of endpoints of this line.

Figure 8.3 shows the result of borders estimated from simulations of parameters by using the information matrix. As can be seen the estimation is quite robust. 100 sets of parameters were simulated and printed in the same picture.

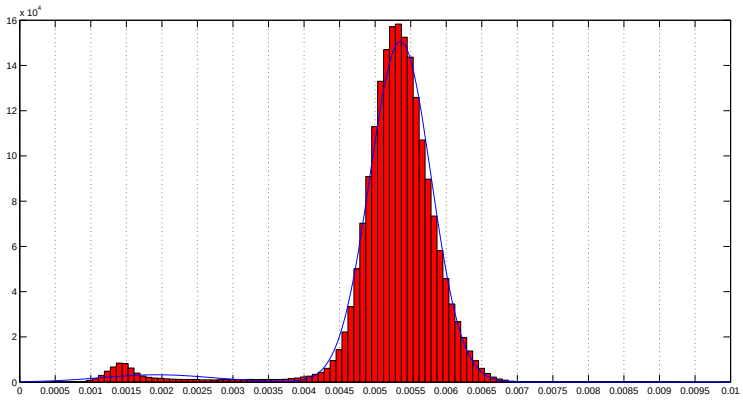

Figure 8.1: Histogram of the values of picture elements of mole. Three dimensional colour elements have been converted to single dimension

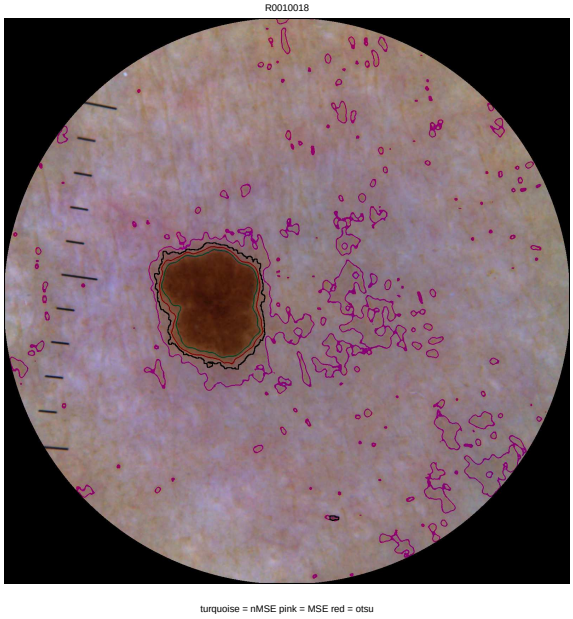

Figure 8.2: Picture of mole with estimated border

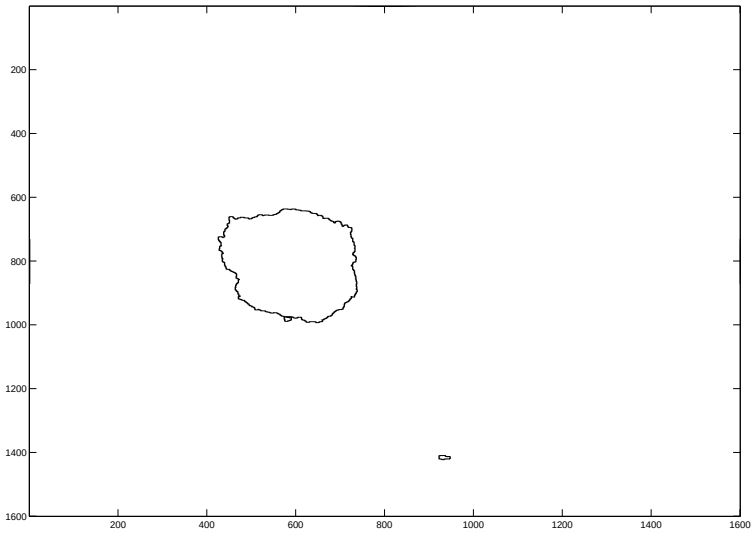

Figure 8.3: Use of the simulation of parameter distribution for testing the robustness of the border estimation.

Appendix A

Appendix

A.1 MATLAB programs

A.1.1 Program for calculating the test statistic

```
%Calculating the test statistic
period_of_stimuli = 18; start_inactivated = 1; end_inactivated = 7;
start_activated = 9; end_activated = 17;
load fmri2 img;  img(:, :, :, 1: 3) = [];
[x_max y_max z_max t_max] = size(img);
complete_periods = floor(t_max / period_of_stimuli);
start_incomplete_period = complete_periods * period_of_stimuli + 1;
points_in_incomplete_period = mod(t_max, period_of_stimuli);
img(:, :, :, 1: points_in_incomplete_period) =...
    img(:, :, :, 1: points_in_incomplete_period)...
    + img(:, :, :, start_incomplete_period: start_incomplete_period ...
+ points_in_incomplete_period - 1); pack;
%periods_indexed = reshape(img(:, :, :, 1: start_incomplete_period - 1), ...
 [x_max y_max z_max period_of_stimuli complete_periods]);
periods_averaged = sum(reshape(img(:, :, :, 1: start_incomplete_period - 1), ...
[x_max y_max z_max period_of_stimuli complete_periods]), 5);
%periods_indexed = [];
periods_averaged(:, :, :, 1: points_in_incomplete_period) =...
    periods_averaged(:, :, :, 1: points_in_incomplete_period) / ...
 (complete_periods + 1);
periods_averaged(:, :, :, points_in_incomplete_period + 1: period_of_stimuli) =...
    periods_averaged(:, :, :, points_in_incomplete_period + 1: period_of_stimuli) ...
/ complete_periods;
%img(:, :, :, start_incomplete_period: start_incomplete_period ...
+ points_in_incomplete_period - 1) = [];
weight_of_times = zeros(1, period_of_stimuli);
if start_inactivated <= end_inactivated
```

```
    weight_of_times(start_inactivated: end_inactivated)...
        = -1 / (end_inactivated + 1 - start_inactivated);
else
    weight_of_times(1: end_inactivated)...
        = -1 / (end_inactivated + period_of_stimuli + 1 - start_inactivated);
    weight_of_times(start_inactivated: period_of_stimuli)...
        = -1 / (end_inactivated + period_of_stimuli + 1 - start_inactivated);
end
if start_activated <= end_activated
    weight_of_times(start_activated: end_activated)...
        = 1 / (end_activated + 1 - start_activated);
else
    weight_of_times(1: end_activated)...
        = 1 / (end_activated + period_of_stimuli + 1 - start_activated);
    weight_of_times(start_activated: period_of_stimuli)...
        = 1 / (end_activated + period_of_stimuli + 1 - start_activated);
end
test_stat = zeros(x_max, y_max, z_max);
for x = 1: x_max,
    for y = 1: y_max,
        for z = 1: z_max,
            test_stat(x, y, z) =...
                weight_of_times * squeeze(periods_averaged(x, y, z, :));
        end
    end
end
%test statistic calculated
%single_dim = reshape(activated_minus_inactivated, ...
    %[x_max * y_max * z_max 1]);
%[sorted_differences, indices_differences] = sort(single_dim);
h_axis_left_end = -30; h_axis_right_end = 62; bin_width = 0.25;
bin_edges = [h_axis_left_end: bin_width: h_axis_right_end];
bin_centers = bin_edges + bin_width/2;
histc_test_stat = histc(test_stat(:), bin_edges);
figure; bar(bin_centers, histc_test_stat, 1, 'w');
%plot histogram of all calculated tests statistics
position0 = round(-h_axis_left_end / bin_width);
histc_noise = histc_test_stat;
for k = position0: -1: 1, histc_noise(2*position0 + 1 - k) = histc_noise(k); end;
histc_noise(2*position0 + 1: end) = zeros(length(histc_noise) - 2*position0, 1);
figure; bar(bin_centers, histc_noise, 1, 'w');
%plot symmetric histogram based on left slope of main peak
histc_signal = histc_test_stat - histc_noise;
figure; bar(bin_centers, histc_signal, 1, 'w');
%plot difference between histograms
```

```
k = 0; load maske_graa; gray_voxels = sum(maske_graa(:));
test_stat_gray = zeros(1, gray_voxels); xs = zeros(1, gray_voxels); ...
ys = zeros(1, gray_voxels); zs = zeros(1, gray_voxels);
for z = 1: z_max,
    for y = 1: y_max,
        for x = 1: x_max,
            if maske_graa(x, y, z) > 0.5
                k = k + 1;
                test_stat_gray(k) = test_stat(x, y, z);
                noise_gray(k) = variances(x, y, z);
                xs(k) = x; ys(k) = y; zs(k) = z;
            end
        end
    end
end
test_stat_noise_gray = test_stat_gray./sqrt(noise_gray);
%m = k;
[sorted_test_stat_gray, stat_ind] = sort(test_stat_gray);
[sorted_noise_gray, noise_ind] = sort(noise_gray);
[sorted_test_stat_noise_gray, stat_noise_ind] = sort(test_stat_noise_gray);

k = 0; load maske_ikkegraa_begrenset; nongray_voxels = ...
 sum(maske_ikkegraa_begrenset(:));
test_stat_nongray = zeros(1, nongray_voxels); xsnon = zeros(1, nongray_voxels); ...
ysnon = zeros(1, nongray_voxels); zsnon = zeros(1, nongray_voxels);
for z = 1: z_max,
    for y = 1: y_max,
        for x = 1: x_max,
            if maske_ikkegraa_begrenset(x, y, z) > 0.5
                k = k + 1;
                test_stat_nongray(k) = test_stat(x, y, z);
                noise_nongray(k) = variances(x, y, z);
                xsnon(k) = x; ysnon(k) = y; zsnon(k) = z;
            end
        end
    end
end
test_stat_noise_nongray = test_stat_nongray./sqrt(noise_nongray);
[sorted_test_stat_nongray, stat_indnon] = sort(test_stat_nongray);
[sorted_noise_nongray, noise_indnon] = sort(noise_nongray);
[sorted_test_stat_noise_nongray, stat_noise_indnon] = ...
sort(test_stat_noise_nongray);



distrib_hist = histc(test_stat_gray, bin_edges);
```

```
figure; bar(bin_centers, distrib_hist, 1, 'w');
histc_noise_gray = distrib_hist;
for k = position0: -1: 1, histc_noise_gray(2*position0 + 1 - k) = ...
 histc_noise_gray(k); end;
histc_noise_gray(2*position0 + 1: end) = zeros(length(histc_noise_gray) ...
 - 2*position0, 1);
histc_signal_gray = distrib_hist - histc_noise_gray;
figure; bar(bin_centers, histc_signal_gray, 1, 'w');
%plot difference between histograms
distrib_hist = histc(test_stat_nongray, bin_edges);
figure; bar(bin_centers, distrib_hist, 1, 'w');

img = []; load fMRI2; img(:, :, :, 1: 3) = [];
gray_size = length(stat_ind); sorted_ascending = zeros(gray_size, t_max);
sorted_ascending_test_statistic = zeros(gray_size, 1);
coordinates_sort = zeros(gray_size, 3);
for l = gray_size: -1: 1,
    sorted_ascending(gray_size - l + 1, :) = reshape(img(xs(stat_ind(l)), ...
 ys(stat_ind(l)), ...
zs(stat_ind(l)), :), [1 t_max]);
    sorted_ascending_test_statistic(gray_size - l + 1) = sorted_test_stat_gray(l);
    coordinates_sort(gray_size - l + 1, :) = ...
[xs(stat_ind(l)) ys(stat_ind(l)) zs(stat_ind(l))];
end
best_detection = test_stat_gray./(51/1296*sqrt(variances_gray));
sorted_best_detected_series = zeros(gray_size, t_max); ...
 sorted_best_detected_statistic = zeros(gray_size, 1);
sorted_best_detected_coord = zeros(gray_size, 3);
[best_detection_sorted, coord_best] = sort(best_detection, 'descend');
for l = 1: gray_size,
    sorted_best_detected_series(l, :) = reshape(img(xs(coord_best(l)), ...
ys(coord_best(l)), zs(coord_best(l)), :), [1 t_max]);
    sorted_best_detected_statistic(l) = test_stat(xs(coord_best(l)), ...
ys(coord_best(l)), zs(coord_best(l)));
    sorted_best_detected_coord(l, :) = ...
[xs(coord_best(l)) ys(coord_best(l)) zs(coord_best(l))];
end

%calculation of average of time series
lower_lin_ind = find(sorted_test_stat_gray > 12, 1)
voxel_count = 100; series_averaged = zeros(1, t_max);
upper_lin_ind = lower_lin_ind + voxel_count - 1;
for l = lower_lin_ind: upper_lin_ind,
    series_averaged = series_averaged + ...
 squeeze(img(xs(stat_ind(l)), ys(stat_ind(l)),
zs(stat_ind(l)), :))';
```

```
end
test_statistic_lower = test_stat(xs(stat_ind(lower_lin_ind)), ...
ys(stat_ind(lower_lin_ind)),
zs(stat_ind(lower_lin_ind)))
test_statistic_upper = test_stat(xs(stat_ind(upper_lin_ind)), ...
ys(stat_ind(upper_lin_ind)), ...
zs(stat_ind(upper_lin_ind)))
series_averaged = series_averaged/voxel_count;
series_averaged_period = zeros(1, 18);
for t = 1: 6,
    series_averaged_period(t) = series_averaged(t) + series_averaged(t + 18) ...
+ series_averaged(t + 36)...
    + series_averaged(t + 54) + series_averaged(t + 72) + series_averaged(t + 90) ...
+ series_averaged(t + 108);
end
for t = 7: 18,
    series_averaged_period(t) = series_averaged(t) + series_averaged(t + 18) ...
+ series_averaged(t + 36) + series_averaged(t + 54) + ...
series_averaged(t + 72) + series_averaged(t + 90);
end
series_averaged_period(1: 6) = series_averaged_period(1: 6)/7;
series_averaged_period(7: 18) = series_averaged_period(7: 18)/6;
figure; plot(1: t_max, series_averaged, '-o'); hold on; ...
plot(1: t_max, mean(series_averaged)*ones(1, t_max), 'r'); hold off;
figure; plot(1: 18, series_averaged_period, '-o'); hold on; ...
plot(1: 18, mean(series_averaged_period)*ones(1, 18), 'r'); hold off;
%img = [];
```

A.1.2 Program for calculating the noise

```
load fmri2 img; img(:, :, :, 1: 3) = []; fMRI_recording = img; img = [];
%load fmri Y; fMRI_recording = Y; Y = [];
%fMRI_recording = sqrt(sigma2)*randn(96, 96, 29, 114);
period_of_stimuli = 18;
[x_max y_max z_max t_max] = size(fMRI_recording);
complete_periods = floor(t_max / period_of_stimuli);
%calculates number of periods
start_incomplete_period = complete_periods * period_of_stimuli + 1;
%start of incomplete sample
points_incomplete_period = mod(t_max, period_of_stimuli); ...
%number of points comprising incomplete period

fMRI_recording = single(fMRI_recording); %reduce storage space for recording
Y2 = fMRI_recording(:, :, :, start_incomplete_period: t_max); ...
Y1 = fMRI_recording(:, :, :, 1: start_incomplete_period - 1);
```

```
%Y2 = reshape(Y2, [x_max y_max z_max 10 1]);
Y1 = reshape(Y1, [x_max y_max z_max period_of_stimuli complete_periods]);
Y3 = Y1(:, :, :, 1: points_incomplete_period, :);
Y3(:, :, :, :, complete_periods + 1) = Y2; %Y3 contains phase with extra samples
Y2 = Y1(:, :, :, points_incomplete_period + 1: period_of_stimuli, :); clear Y1; pack;
%Y2 contains phase with no extra samples
variances = single(zeros(x_max, y_max, z_max, period_of_stimuli));
%specify dimension of matrix
variances(:, :, :, 1: points_incomplete_period) = ...
var(Y3(:, :, :, :, :), 0, 5) * complete_periods; clear Y3;
%calculates sample variances for phase with extra samples
variances(:, :, :, points_incomplete_period + 1: period_of_stimuli) ...
= var(Y2(:, :, :, :, :), 0, 5) * (complete_periods - 1); clear Y2;
%calculates sample variances for phase with no extra samples
variances_series = sum(variances(:, :, :, :), 4) /...
    (complete_periods * points_incomplete_period + (complete_periods - 1) ...
* (period_of_stimuli - points_incomplete_period));
%pools the variances to obtain estimate of variance for time series
variances = double((points_incomplete_period * (complete_periods + 1) + ...
(period_of_stimuli - points_incomplete_period) * complete_periods) /...
    (points_incomplete_period * (complete_periods + 1) * ...
(period_of_stimuli - points_incomplete_period) * complete_periods) ...
* variances_series);

k = 0; [x_max y_max z_max] = size(variances_series); load maske_graa; ...
gray_voxels = sum(maske_graa(:));
variances_gray = zeros(1, gray_voxels); xs = zeros(1, gray_voxels); ...
ys = zeros(1, gray_voxels); zs = zeros(1, gray_voxels);
for z = 1: z_max,
    for y = 1: y_max,
        for x = 1: x_max,
            if maske_graa(x, y, z) > 0.5
                k = k + 1;
                variances_gray(k) = variances_series(x, y, z);
                xs(k) = x; ys(k) = y; zs(k) = z;
            end
        end
    end
end
[sorted_variances, coord] = sort(variances_gray);
%variances_gray(k + 1: end) = [];
%xs(k + 1: end) = []; ys(k + 1: end) = []; zs(k) = [];
%figure; hist(variances_gray, 2000);
bin_edges_var = 0: 2.5: 8000; bin_centers_var = bin_edges_var + 1.25;
var_distrib = histc(variances_gray, bin_edges_var);
figure(2); bar(bin_centers_var, var_distrib, 1, 'w');
```

A.1.3 Program for restoring fMRI images

```
global x y z neighbours lower_x lower_y lower_z...
    upper_x upper_y upper_z beta activated_minus_inactivated estimated_image...
    variances twice_sigma_squared;
lower_x = 1; lower_y = 1; lower_z = 1;
upper_x = 96; upper_y = 96; upper_z = 29;

    neighbours = [0 0 1 1; 0 1 0 1; 1 0 0 1; 0 0 -1 1; 0 -1 0 1; -1 0 0 1;...
    0 1 1 1 / sqrt(2); 1 0 1 1 / sqrt(2); 1 1 0 1 / sqrt(2); 0 1 -1 1 / sqrt(2);...
    1 0 -1 1 / sqrt(2); 1 -1 0 1 / sqrt(2); 0 -1 1 1 / sqrt(2); -1 0 1 1 / sqrt(2);...
    -1 1 0 1 / sqrt(2); 0 -1 -1 1 / sqrt(2); -1 0 -1 1 / sqrt(2); ...
    -1 -1 0 1 / sqrt(2); 1 1 1 1 / sqrt(3); 1 1 -1 1 / sqrt(3); ...
    1 -1 1 1 / sqrt(3); -1 1 1 1 / sqrt(3); ...
    1 -1 -1 1 / sqrt(3); -1 1 -1 1 / sqrt(3); ...
    -1 -1 1 1 / sqrt(3); -1 -1 -1 1 / sqrt(3)];
betas = single([0.5 1 1 0.5 1 1 1 / sqrt(2) 1 / sqrt(2) 1 / sqrt(2) ...
    1 / sqrt(2) 1 / sqrt(2) 1 / sqrt(2)
    1 / sqrt(2) 1 / sqrt(2) 1 / sqrt(2) 1 / sqrt(2) 1 / sqrt(2) 1 / sqrt(2) ...
    1 / (2 * sqrt(2)) 1 / (2 * sqrt(2)) 1 / (2 * sqrt(2)) 1 / (2 * sqrt(2)) ...
    1 / (2 * sqrt(2)) 1 / (2 * sqrt(2)) 1 / (2 * sqrt(2)) 1 / (2 * sqrt(2))]);
neighbours(:, 4) = betas';

beta = 0.7;
for iteration = 1: 10,
    tic
    previous_estimated_image = estimated_image;
for z = lower_z: upper_z,
    intermediate_k_pl_1(:, :, 1) = estimated_image(:, :, z);

    %ICM-algoritme for snitt i xy-retning for gitt z
    for y = lower_y: upper_y,
        for x = lower_x: upper_x,
            %ICM-algoritme for enkelt voxel
            sigma = sqrt(variances(x, y, z));
            half_interval_width = 5 * sigma; twice_sigma_squared = 2 * sigma ^ 2;
            intermediate_k_pl_1(x, y, 1)...
                = fminbnd(@potential_U3, estimated_image(x, y, z) ...
                - half_interval_width,
                estimated_image(x, y, z) + half_interval_width);
            %(slutt p ICM-algoritme for enkelt voxel, resultat i
            %intermediate_k_pl_1(x, y, 1))

        end
        %(slutt p ICM-algoritme for gitt y og z, resultat i
        %intermediate_k_pl_1(:, y, 1))
```

```
    end
    if z > lower_z estimated_image(:, :, z - 1) = ...
            intermediate_k_pl_1(:, :, 2); end
    intermediate_k_pl_1(:, :, 2) = intermediate_k_pl_1(:, :, 1);
    %(slutt p ICM-algoritme for gitt z, resultat i
    %intermediate_k_pl_1(:,:, 1)

end
estimated_image(:, :, upper_z) = intermediate_k_pl_1(:, :, 2);
%(slutt p iterasjon i ICM-algoritme, oppdatert estimat etter siste
%iterasjon i estimated_image
toc
iteration
end
```

Bibliography

Larsen, R. J. and Marx, M. L.: An Introduction to Mathematical Statistics and Its Applications. Third Edition. Prentice-Hall, 2001.

Casella, G. and Berger, R. L.: Statistical Inference. Second Edition. Duxbury, 2002.

Besag, J.: On the Statistical Analysis of Dirty Pictures. Journal of the Royal Statistical Society. Series B (Methodological), Vol. 48, No. 3. (1986), pp. 259-302.

Robert, C. P. and Casella, G.: Monte Carlo Statistical Methods. Second Edition. Springer, 2004.

Stuart, A., Ord, J. K. and Arnold, S: Kendall's advanced Theory of Statistics. Volume 2A: Classical Inference and the Linear models. Arnold, 1991.